CD-ROM付

Windows95/98/NT4対応

10日でおぼえる Visual Basic 6.0 入門教室

VBテックラボ&瀬戸 遥 著

SE
SHOEISHA

本書の内容

本書は、Visual Basic 6.0を使って、はじめてWindowsのプログラムを作成しようとする人や、他のプログラミング言語を使ったことはあるが、Visual Basicを使うのははじめて、という方を対象に、すぐにVisual BasicでWindowsのプログラムを作成できるように、随所に工夫を凝らした構成で編集しました。

全体の流れは、「プログラミングセミナー」のように、段階を追ってVisual Basicの使い方やテクニックを説明してあります。各レッスンは、4段階になっています。

【レッスンの構成】

1. サンプルの説明
サンプルの動作と、
作成の際のポイントを解説します。

2. 操作説明
作成に伴う一連の操作を、
わかりやすいように画面を使って
説明します。

3. コードの解説
プログラムコードを、
丁寧に解説していきます。

4. 練習問題
練習問題で基礎を固め、
応用力をつけます。

紹介しているサンプルのコードは、入門者が混乱しないように、説明のポイントを重点的に絞って、なるべく少ない行数で作成しました。もっとも長いものでもコードの入力に時間を取られることはありません。また、これらのサンプルはコンパクトで理解しやすく、様々なテクニックを簡潔に紹介しているので、自分で開発する際に応用することができます。サンプルの完成バージョンは付属CD-ROMに収録されています。

本書では、従来のプログラミング解説書のように「Visual Basicとはなにか」というその概要や概念の説明の長さで、プログラムを作る前からうんざりしてしまうようなことはありません。そういった「解説」はレッスンの最後にまとめて、まず「作ること」それ自体を楽しんでもらうことを主としたレッスン構成にしています。

本書は「作りながらおぼえたい人」と「おぼえてから作りたい人」のどちらにもわかりやすいように書かれています。

10日でおぼえる Visual Basic 6.0 入門教室

VBテックラボ&瀬戸 遥 著

SE
SHOEISHA

本書内容に関するお問い合わせについて

このたびは翔泳社の書籍をお買い上げいただき、誠にありがとうございます。弊社では、読者の皆様からのお問い合わせに適切に対応させていただくため、以下のガイドラインへのご協力をお願い致しております。下記項目をお読みいただき、手順に従ってお問い合わせください。

●ご質問される前に

弊社Webサイトの「正誤表」や「出版物Q&A」をご確認ください。これまでに判明した正誤や追加情報、過去のお問い合わせへの回答（FAQ）、的確なお問い合わせ方法などが掲載されています。

正誤表　　http://www.seshop.com/book/errata/

出版物Q&A　　http://www.seshop.com/book/qa/

●ご質問方法

弊社Webサイトの書籍専用質問フォーム（http://www.seshop.com/book/qa/）をご利用ください（お電話や電子メールによるお問い合わせについては、原則としてお受けしておりません）。

※質問専用シートのお取り寄せについて

Webサイトにアクセスする手段をお持ちでない方は、ご氏名、ご送付先（ご住所／郵便番号／電話番号またはFAX番号／電子メールアドレス）および「質問専用シート送付希望」と明記のうえ、電子メール（qaform@shoeisha.com）、FAX、郵便（80円切手をご同封願います）のいずれかにて“編集部読者サポート係”までお申し込みください。お申し込みの手段によって、折り返し質問シートをお送りします。シートに必要事項を漏れなく記入し、“編集部読者サポート係”までFAXまたは郵便にてご返送ください。

●回答について

回答は、ご質問いただいた手段によってご返事申し上げます。ご質問の内容によっては、回答に数日ないしはそれ以上の期間を要する場合があります。

●ご質問に際してのご注意

本書の対象を越えるもの、記述個所を特定されないもの、また読者固有の環境に起因するご質問等にはお答えできませんので、予めご了承ください。

●郵便物送付先およびFAX番号

送付先住所　〒160-0006　東京都新宿区舟町5

FAX番号　03-5362-3818

宛先　（株）翔泳社出版局 編集部読者サポート係

本書で紹介した全てのレッスンは、マイクロソフト株式会社より発売中のMicrosoft® Visual Basic® 6.0 Professional Edition及びEnterprise Edition上で動作するように作成されています。Visual Basic 6.0 Learning Editionでは、動作可能なものもありますが一部のレッスンは動作しません。各レッスンにおけるVisual Basic 6.0 Learning Editionでの動作の可否については、本文中の各レッスンの巻頭ページ左下に以下のように明記していますのでご確認ください。

なお、本書にはVisual Basic本体は収録されておりません。Visual Basicをお持ちでない方は別途ソフトが必要です。

◆記号の意味

◎…動作可能　×…動作不可

◆使用例（動作不可の場合）

× Learning Edition

ようこそ、Visual Basic 6.0 入門教室へ！

この授業をすべて受講すれば、あなたもVisual BasicでWindowsのプログラムが簡単に作れるようになります。

これまでかなり専門的な知識がないと作成できなかったWindowsのプログラムが、Microsoft® Visual Basic®（以下Visual Basic）の登場でだれでも気軽に作成できるようになりました。

Visual Basicは、Microsoftから発売されている、Windows95/98/Windows NT4.0で動作する32ビットアプリケーションを作成するプログラミングソフトです。マウスの操作だけで、簡単にプログラムの外観を作成できる手軽さと、Basic言語をベースにした学習しやすさを兼ね備えた、大変わかりやすいプログラム言語です。

わかりやすい言語である一方で、Windowsの持っている多くの機能を、自分のプログラムに取り入れることができる、強力な開発環境になっています。しかし、それでも、Windows独特の「イベント対応」プログラミングを習得するには、多少の努力が必要になっています。

本書では、このVisual Basicの言語仕様やプログラミング概念を、特にわかりやすく解説しています。また、フォームやコントロール、プロパティとメソッドなどについても、実際にプログラムを作成しながら覚えていく学習スタイルを採用し、一段とわかりやすい解説書に仕上げています。

１つのレッスンに１つの例題プログラムを使用し、基本的なコントロールを使う初歩的なテクニックから、マルチメディアやデータベースを扱う中級レベルのテクニックまで、実用的なテクニックを幅広く紹介しています。Lesson1から最後まで学習していけば、必ずVisual Basicを使ってWindowsのプログラムが作成できるように、カリキュラムを組んでいますので、またたくまにVisual BasicによるWindowsのプログラミングテクニックをマスターできるでしょう。

ぜひ、本書を使って、Visual Basicプログラミングを楽しんでください。

1998年11月1日　瀬戸 遥

目次

ようこそ、Visual Basic 入門教室へ！ iii

オリエンテーション ix

Visual Basic へようこそ x
- Visual Basic とは x
- Visual Basic のひろがる世界 x
- Visual Basic 6.0 でできること xi

Visual Basic のプログラミングとは xii
- Visual Basic と他の開発言語との違い xii
- Windows アプリケーションとは xiii
- Visual Basic は「旅行会社」？ xiv
- イベント駆動型プログラミング xv
- 視覚的・対話的な機能による効率的な開発 xvi
- 最低限知っておきたい用語 xvii

プログラミングをはじめる前に xviii
- 本書の構成 xviii
- レッスン前のチェックポイント xxi
- CD-ROM の使い方 xxi

Visual Basic の開発環境 xxii
- Visual Basic の開発環境 xxii

Visual Basic の画面構成 xxviii
- Visual Basic の各部の名称 xxviii

Visual Basic の基本操作 xxx

CD-ROM をご使用になる前に xxxii

第1日 はじめてのプログラムを作る 1

1時限目 Lesson 1 コマンドボタンでメッセージを表示する 2
Lesson 2 終了ボタンを追加する 14
2時限目 Lesson 3 プロパティを操作する 22
Lesson 4 プロパティへの値の代入 30
3時限目 Lesson 5 イメージファイルを使う 36
Lesson 6 イメージを切り替える 42
4時限目 Lesson 7 コントロールを組み合わせる 46

第2日 イベントを使う 55

1時限目 Lesson 8 クリックとダブルクリック 56
2時限目 Lesson 9 マウスイベントを使う 62
3時限目 Lesson 10 変数を使う 70
4時限目 Lesson 11 MouseDownとMouseUpの組み合わせ 76
5時限目 Lesson 12 お絵描きプログラムの作成 82

第3日 メニューを使う 97

1時限目 Lesson 13 メニューを組み込む 98
2時限目 Lesson 14 右ボタンポップアップメニューを作る 104
3時限目 Lesson 15 メニューを追加する 110
4時限目 Lesson 16 メニューの削除 118
5時限目 Lesson 17 状況感知型メニューを作成する 124

第4日 コントロールの連携と動作 137

1時限目 Lesson 18 アニメーション表示プログラム 138
2時限目 Lesson 19 スクロールバーとツールチップを使う 160
3時限目 Lesson 20 チェックボックス・リストボックスコントロールを使う 174
4時限目 Lesson 21 コンボボックスでのリスト項目の追加と削除 186

第5日 **ファイルを操作する** **197**

1時限目 Lesson 22 テキストファイルを読み込む 198
2時限目 Lesson 23 リスト項目をファイルに書き出す 210
3時限目 Lesson 24 イメージビューワーの作成 220

第6日 **ソフトウェアランチャーの作成** **231**

1時限目 Lesson 25(1) メインモジュールの作成 232
2時限目 Lesson 25(2) セットアップモジュールの作成 258

第7日 **いろいろなテクニック** **271**

1時限目 Lesson 26 コモンダイアログボックス 272
2時限目 Lesson 27 マルチメディアを使う 282
Lesson 28 コードでサウンドファイルを再生する 288
3時限目 Lesson 29 データコントロール 292
4時限目 Lesson 30 タブ付きダイアログボックス 298
5時限目 Lesson 31 フォームテンプレートの活用 308

第8日 **デバッグとエラー処理** **317**

1時限目 Lesson 32 変数のウォッチを使う 318
2時限目 Lesson 33 ブレークポイントを使う 324
3時限目 Lesson 34 エラー処理を使う 328

第9日 **実行ファイルの作成と配布** **337**

1時限目 Lesson 35 EXEファイルの作成 338
2時限目 Lesson 36(1) バージョンと作成情報を設定する 342
3時限目 Lesson 36(2) ディストリビューションウィザード 348

第10日 ActiveXコントロールを作る 361

ActiveXコントロールの概要 362
1時限目 Lesson 37(1) スロットマシンコントロールの作成 364
2時限目 Lesson 37(2) ActiveXコントロールの組み込みとテスト 374
3時限目 Lesson 38 ActiveXコントロールをHTMLに組み込む 380

付録 各種リファレンス 391

用語集 392
関数リファレンス 398
ASCII文字コード一覧 408
データ型一覧 409
エラー番号一覧 410
プロジェクトファイルの形式 412
ショートカット一覧 412
練習問題の解答 415

索引 433

Let's VBSchool!!

オリエンテーション

Visual Basicを使ってWindowsのプログラムを作成するには、Visual Basicがどのようなもので、どういう操作を行っていけばよいのかを知っておく必要があります。そこで、実際のプログラム作成に入る前に、オリエンテーションを兼ねて予備知識を身に付けておきましょう。

今日のレッスンで解説する項目

- ❏ Visual Basicとは
- ❏ Visual Basicのひろがる世界
- ❏ Visual Basic 6.0でできること
- ❏ Visual Basicと他の開発言語との違い
- ❏ Windowsアプリケーションとは
- ❏ イベント駆動型プログラミング
- ❏ 視覚的・対話的な機能による効率的な開発
- ❏ 最低限知っておきたい用語
- ❏ 本書の構成
- ❏ レッスン前のチェックポイント
- ❏ CD-ROMの使い方
- ❏ Visual Basicの開発環境
- ❏ Visual Basicの画面構成
- ❏ Visual Basicの基本操作

Visual Basic へようこそ

Microsoft Visual Basic6.0 は、だれでも簡単にWindows95/98/Windows NTのプログラムを作成できる、32ビットプログラム開発ソフトです。ここでは、Visual Basicでどういうことができるのか、説明しておきます。

Visual Basicとは

Visual Basicは、Windowsのプログラムを開発する目的で作成されたプログラム作成ソフト（プログラム開発言語）です。Windows 3.1の時代に最初のバージョンがリリースされましたが、それまでWindowsのプログラムを作成するには、C言語かPascal（パスカル）言語で、Windowsの「API（Application Programming Interface)」を呼び出すなどで、アプリケーションを作成していました。

「API」というのはOS（Windows）が用意しているアプリケーション用の共通コマンドの集まりですが、これを扱うことは、プログラマにとって大変煩雑な作業で、複雑なコードの記述とWindowsに対するかなり高度な知識を求められていました。

そこで、従来から定評のあったコンピュータ学習用プログラミング言語「Basic（Beginner's All-purpose Symbolic Instruction Code)」をベースに、Windowsのプログラムをだれでも簡単に作成できるように改良して作られたのが、Visual Basicです。

Windowsの外観（ユーザーインターフェイス）を、コントロールと呼ばれる部品を使って作成することで、見たままのユーザーインターフェイスを画面上に組み立てることができます。また、WindowsのAPIも、コントロールやメソッド、プロパティなどに割り当てることで、より直感的でわかりやすい記述を使って、プログラムを作成できるようになっています。

Visual Basic 6.0の開発画面

Visual Basicのひろがる世界

Visual Basicは、当初は単独で動作するプログラムの作成が主な機能でしたが、バージョンを重ねるごとにその機能を強化拡張してきました。Windowsの進化とともに、通信

機能やデータベースへのアクセス機能、インターネット機能などを組み込み、これらに対応したプログラム開発ができるようになりました。

Microsoftの最新のテクノロジーである「Active Xコントロール」の作成機能も装備されています。複雑な機能をコントロール（部品）化することで、再利用可能な汎用性の高いプログラムを作成可能です。

バージョン6.0では「DHTML」や「WebClass」と呼ばれるコンポーネント作成までもサポートし、Visual Basicのアプリケーションを作るのと同じ感覚でダイナミックなWebコンテンツを開発できるようになりました。また、データベースと連携した高度なアプリケーション開発の機能も強化されています。

ActiveXコントロールを作成できる

Visual Basic 6.0でできること

Visual Basicには、標準コントロールと呼ばれる、コマンドボタンやリストボックスなどのWindowsの標準機能をコントロール化したものが装備されています。これらのコントロールを使うことで、普段のWindowsのアプリケーションと同じような外観と機能を作成できるようになっています。プログラムからファイルを開いたり、保存したりすることも自由ですし、グラフィック関係のメソッドを使うことで、お絵描きプログラムを簡単に作成したりすることができます。その他にも、以下のような多彩な機能を提供する「ActiveX」が拡張機能として組み込まれています(Professional Edition以上)。

- 「マルチメディアコントロール」を使えば、サウンドファイルの再生やAVIビデオの再生、CD-ROMドライブの制御など、マルチメディア機能を組み込めます。
- 「データコントロール」を使えば、Accessなどのデータベースファイルに容易にアクセスし、データの読み書きができます。
- 「コミュニケーションコントロール」を使うと、シリアルポートを通じて通信機能を装備できます。
- 「インターネットトランスファコントロール」は、インターネットで使用されているハイパーテキスト転送プロトコル（HTTP）や、ファイル転送プロトコル（FTP）の機能を提供します。

マルチメディアコントロール

データコントロール

このように、コントロールと関連するプロパティ、メソッド、イベントを組み合わせて、Windowsの持っている多くの機能を、Visual Basicのプログラムに組み込むことができます。Visual Basicを学習することによって、その知識や技術をWindows用ソフト開発の様々な分野に応用することができるのです。

Visual Basicのプログラミングとは

ここでは、Visual Basicのプログラム開発の特徴を、他の言語と比較しながら紹介します。多彩な機能をプログラムに簡単に組み込んで開発できるVisual Basicですが、「簡単」とはいえ、ある程度プログラミングにまつわる知識は必要です。

Visual Basicと他の開発言語との違い

Visual Basicは、コンピュータの動作原理を学習するために開発されたBasic言語をベースに作成されています。しかし、NEC社のN88-BasicやMicrosoft社のMS-DOS用Basicと違い、Windowsのプログラム開発を目的として作成し直されており、プログラミングスタイルはずいぶん違うものになっています。

●ユーザーインターフェイスの作成が簡単

従来のBasicでは、画面の作成はすべてコードで行っていました。例えば、プログラムにボタンを追加したいときは、まずそのボタンをコードによって描画しなければなりませんでした。

Visual Basicでは、ユーザーインターフェイスの作成はすべて部品化されています。ボタンを付けたければ、ツールボックスからボタンの「部品」をマウスを使ってドラッグするだけで組み立てることができます。

●処理ごとにコードを別々の場所に記述

従来のBasicではプログラムの起動から終了までのコードを、1つのファイルに、動作の順番に記述していました。場所によってはラベルを使ってサブルーチンを組み立て呼び出すスタイルを取っていました。

Visual Basicでは、プログラムの動作コードは、各コントロールやフォームの「イベントプロシージャ」と呼ばれる場所に記述し、ユーザーの操作に対応する形式で処理を組み立てていきます。ファイルの構成も、「プロジェクト」という管理形態でソースコードや画面（フォーム）を保存します。

従来のBasic言語と比べて、大幅に変わっている部分はありますが、それはプログラムが動作する環境がウィンドウシステムになったことが大きく、言語としてはより記述がしやすく、理解しやすいように改良されているのが特徴です。

ステートメントなどは従来のBasic言語と同じ機能や名前を継承している部分もありますので、Windowsのプログラムを開発するために、C/C++言語やPascal言語を最初から習得するよりも、短時間でマスターできるメリットを持っています。

Windowsアプリケーションとは

Windowsのプログラムは、いったいどのように動いているのでしょう？ 特徴は大きく2つあります。

1つ目は、Windowsでは、常にWindowsのシステムとプログラム（アプリケーション）が「メッセージ」を通信しながら動作しています。例えば、Excelのセルをクリックすると、「ユーザーがマウスのボタンをクリックした」という情報は、Windowsのシステムが把握します。そして、Excelに対して「ユーザーがマウスのボタンをクリックした」という情報を通知します。「マウスクリック」の通知を受け取ったExcelは「セルを選択する」「メニューを表示する」ということを状況に応じて割り振り、ユーザーに結果を表示します。

アプリケーションの動作：Excelの場合

初期状態

①ユーザーがマウスをクリックする

イベント発生！

メッセージA：「マウスがクリックされた！」

Windows 95

②「マウスがクリックされた！」というメッセージAをOSが取得

メッセージ！

③WindowsからExcelへメッセージAが通知

Excel

④「セル上でマウスがクリックされたらセルを選択」という処理を実行

ユーザーへの結果表示

2つ目は、Windowsには、「API」と呼ばれる基本命令の集まりが用意されています。APIとは、「Apprication Programming Interface」の略で、Windowsの持っている機能を公開し、プログラミング言語から利用できるようにしたインターフェイス機能のことです。これは共通仕様として定まっており、Windowsの上で動作するプログラム（アプリケーション）は、ウィンドウ枠やコマンドボタンなど、すべてこのAPIを利用して作成されているため、APIを利用したアプリケーションは「Windowsらしく」なるというわけです。

典型的なWindowsアプリケーション
①ウィンドウ
②メニュー
③ツールバー
④コンボボックス
⑤スクロールバー
⑥テキストボックス
⑦ダイアログボックス
⑧ステータスバー

Visual Basicは「旅行会社」？

しかし、例えば、APIをC言語で扱うには、Windowsの動作原理や機能についてかなり詳しい知識とともに、APIをC言語から使用する複雑な方法を熟知していないと利用できません。

そこで、Visual Basicでは、コントロールやプロパティ、メソッド、イベントといったシステムを用意し、これらが複雑なAPIへのアクセスを「代行」してくれます。そのため、Visual BasicプログラマがAPIについて詳細を知らなくても、「Windowsらしい」プログラミングができるように設計されています（もちろん熟達したVisual BasicプログラマはAPIをVisual Basicから利用することも可能です）。

これは、海外に行くのにパッケージツアーを利用するのか、個人旅行にするのかを選択するのと似ています。

前者はホテルや航空券などを旅行会社が準備してくれますが、後者はそういったことを自分ですべて手配しなければなりません。もちろん、個人旅行にはそれだけ自由度が大きく身軽という点もありますが、はじめてプログラミングという「海外」へアクセスしようとしているビギナーなら、まずパッケージツアーからはじめてみるのがよいのではないでしょうか。物足りなくなったらいつでも「個人旅行」へ踏み出すことができますし、そのときもツアーでの経験や知識がきっと役に立つはずです。

Visual Basicの場合

Win ホテル / Win 航空 / Win レストラン
代行 / 代行 / 代行
「Visual Basic」旅行会社
交渉
ユーザー（プログラマ）

Visual Basicとわかりやすい言葉で交渉できるのでプログラマの負担が軽い

Cなどの場合

Win ホテル / Win 航空 / Win レストラン
直接交渉 / 直接交渉
C/Pascal
ユーザー（プログラマ）

通訳はいるが「直接交渉」になるのでそれなりのプログラマの技量が必要

イベント駆動型プログラミング

Windowsのプログラミングの特徴の1つに、「イベント駆動型プログラミング」があります。例えば、これまでのBasicプログラムでは、ユーザーの選択をプログラムの側で用意し、ユーザーにどの操作をするのかを強制的に選択させていました。例えばメニューを3つ用意し、「どれかを選んでください」というプログラムの作成スタイルです。

従来のBASICプログラミングの典型例

```
好きなプログラミング言語はどれですか？
    1. Visual Basic
    2. C
    3. Java
(数字キーを押してください)_
```

しかし、Windowsのアプリケーションでは、ユーザーがファイルを開きたいときには、［開く］コマンドを選択するといった様に、自分の好きなときに好きな操作ができるようになっています。したがって、プログラム側としてはユーザーが操作できるインターフェイスをあらかじめいくつか用意しておき、それらのインターフェースに変化があったときに処理を組み立てていきます。

そのために、プログラムの記述も、ユーザーが行う操作（これを「イベント」といいます）にあわせて作成していきます。具体的には、ユーザーがコマンドボタンを押したときには、Aの処理を実行し、メニューが選択されたときはBの処理を実行させる、という記述方法になります。ユーザーの「イベント」によってプログラムが「駆動（動作）」する、つまりこれが、「イベント駆動型プログラミング」と呼ばれているものです。

ユーザーインターフェイスに起こる可能性のある「イベント」は、フォームやコントロールなどのVisual Basicの部品1つ1つにあらかじめ用意されています。これらをうまく組み合わせて、イベントに対するアクションを作成していくのが、Visual Basicのプログラミングスタイルです。

「イベント駆動型」のプログラミング

Command1

イベント発生！

押す

Lesson1

はじめてのプログラムです。

OK

```
Private Sub Command1_Click()
    MsgBox "はじめてのプログラムです。"
End Sub
```

①ユーザーがマウスをクリックする

②コマンドボタンにClickイベントが発生する

③Clickイベントのコードを実行

④処理結果の表示

逆に、用意されていないイベントを扱うことはできませんが、他のイベントが相互にカバーしあっているので不便で困るということはありません。例えば、あり得ないことですが「コマンドボタンを押さなかったとき」というイベントを使いたいと思ったとしても、コマンドボタンにはそういうイベントは用意されていません。しかし「コマンドボタンを押さない」というのは、なにもしない時か、他のボタンやメニューが押されている時ですから、そもそもそういうイベントを用意する必要がないわけです。そういう不必要なイベントは最初から用意されていません。

```
Private Sub Form_MouseUp(Butt
    Dim Message As String

    If Button = 1 Then
        Message = "マウスの位
        MsgBox Message
    End If
End Sub
```

LinkError / LinkExecute / LinkOpen / Load / LostFocus / MouseDown / MouseMove / MouseUp / OLECompleteDrag / OLEDragDrop / OLEDragOver / OLEGiveFeedback

必要なイベントは用意されている

視覚的・対話的な機能による効率的な開発

Visual Basicは、その名のとおりビジュアルにプログラムを開発していく言語です。

ユーザーインターフェイスの構築は、実際に見たままの形で動作できます。イベントに対応するコードの記述も、リストボックスから必要なオブジェクトとイベントを選ぶだけです。プロパティの設定は、プロパティウィンドウにある一覧表示されたリストから値を選んで設定します。定数が決まっているプロパティでは、値そのものがリスト表示され、場合によってはダイアログボックスが開き、対話的に設定できます。このように、プログラマが行うべき操作をわかりやすくして、開発にかける負担を軽くし、効率的なアプリケーション開発ができる環境を提供しています。

Visual Basicによるプログラムの開発手順

①外観をデザインする → ②処理に必要なコードを記述する → ③プログラム実行 → ④デバッグ

最低限知っておきたい用語

Visual Basicを習得するうえで、いくつかの専門的な用語や概念の理解は避けて通れないものです。現時点ではこれらを「完全に理解する」必要はありませんが、「なんとなくわかる」程度には慣れておく必要があるでしょう。詳しくはレッスンの中で、実際にプログラムを作成しながら解説していきますが、予備知識として、ここでは特に重要な専門用語を中心に説明しておきます。

- ●イベント・・・・・・ユーザーが行った操作全般のことです。コマンドボタンを押せば「Click」、マウスを動かせば「MouseMove」というイベントが発生します。
- ●オブジェクト・・・・フォームやコントロールなど、プログラムを形成するすべての要素をオブジェクトと呼びます。
- ●コード・・・・・・・プログラム言語で作成された記述のこと。ソースコードとも呼びます。
- ●コントロール・・・・プログラムの外観や動作を形成する部品をコントロールと呼びます。機能によって、何種類も用意されています。
- ●ステートメント・・プログラムの流れや処理を制御する命令です。
- ●フォーム・・・・・・プログラムウィンドウのベースになるものです。ウィンドウの外観は、このフォームにコントロールを組み込んで作成します。
- ●プロジェクト・・・Visual Basicで作成したプログラムに使用するファイルを管理する形態です。Visual Basicでは、1つのプログラムが複数のファイルで形成されるため、それぞれのファイルの依存関係を把握し、管理するために「プロジェクトファイル」が用意されています。
- ●プロパティ・・・・・コントロールやフォームの属性を設定するために使用されます。多くのコントロールは、たくさんのプロパティを持ち、この値を変えることでコントロールの外観や動作範囲などを設定します。
- ●プロシージャ・・・・サブルーチンのことで、プログラムのコードを役割ごとに小さな固まりにまとめたものです。
- ●メソッド・・・・・・オブジェクトを直接操作するために用意された機能です。
- ●モジュール・・・・・プログラムを小規模なかたまりに分けたものです。プログラムの処理を内容によってモジュール単位に分けることで、コードの管理がしやすくなります。
- ●関数・・・・・・・・式の中で使用するもので、ある値を指定して実行すると、その結果を返してきます。
- ●変数・・・・・・・・データを貯えておく「入れ物」のことです。プログラムの処理の中で、式の結果を一時的に保管する場合などに使用します。

プログラミングをはじめる前に

本書の特徴とCD-ROMの使い方を説明します。

本書の構成

本書は、初めてWindowsのプログラムを作成しようとする読者や、他のプログラミング言語は使ったことがあるがVisual Basicを使うのは初めて、というユーザーを対象に、すぐにVisual BasicでWindowsのプログラムを作成できるように、随所に工夫を凝らした構成にしました。

全体の流れは、「プログラミングセミナー」のように、段階的にVisual Basicの使い方やテクニックを説明し、無理なく習得できるペースに設定しています。サンプルプログラムをレッスンごとに分け、約10日で全体を学べるように構成しました。レッスンは大きく分けて4つに構成されています。

【レッスンの構成】

①サンプルプログラムの説明
　サンプルプログラムの動作とポイントを解説します。

②コントロールやプロパティの設定
　プログラム作成手順を、豊富な画面を使って操作の流れを1つずつ説明します。

③コードの解説
　プログラムのコードを1行ずつ解説していきます。

④まとめと練習問題
　レッスンのまとめと練習問題を付けました。

紹介しているサンプルプログラムのコードは、入門者が混乱しないように、説明のポイントを重点的にしぼって、なるべく少ない行数で作成しました。これらのサンプルはコンパクトで理解しやすいので、自分でプログラムを開発する際に応用することができます。

本書では、Windowsのソフトを使い慣れている人であればおそらくどなたでも、Lesson1からすぐにVisual Basicのプログラムを作りはじめることができるでしょう。従来のプログラミング解説書のように「Visual Basicとはなにか」というその概要や概念の説明を延々と読まされ、プログラムを作る前からうんざりしてしまうようなことはありません。そういった抽象的な概念は重要ですが、かといってそればかりに足を取られていても本末転倒です。本書では、そういった「解説」はレッスンの最後にまとめて、まず「作ること」を楽しんでもらうことを主としたレッスン構成にしています。

ちょっと複雑な理屈やVisual Basic独特のプログラミング作法にとまどうことがあるかもしれません。もしそうなっても、とりあえずは後回しにして、実際にプログラムをどんどん作りながら前進していってください。Visual Basicの世界に慣れていけば知識も自然と身に付くはずです。本書は「作りながらおぼえたい人」と「おぼえてから作りたい人」のどちらにもわかりやすいように書かれています。

①サンプルプログラムの説明およびコントロールとプロパティの設定

Lesson 17

第3日
5時限目

メニューを使う

Lesson17.vbp
VB6CD
Lesson17

Learning Edition

状況感知型メニューを作成する

メニューコントロールの［Visible］プロパティを使う方法をマスターします。

● 今回作成するプログラム

1 右クリックでポップアップメニューが表示される

2 ［表示］－［フォームの表示］をクリックすると…

3 Form2が表示される

4 右クリックでポップアップメニューが表示される

POINT

Windows 95のアプリケーションでよく使われているポップアップメニューは、状況に応じてメニュー項目が変化します。
配置したメニューコントロールの［Visible］プロパティを使うと、この状況に応じて変化するポップアップメニューを作成できます。
このプログラムは、ポップアップメニューのときだけ専用のメニューを表示するプログラムです。2つのフォームを持ち、それぞれが別のポップアップメニューを表示します。

コントロールとプロパティの設定

※新しいプロジェクトとフォームを用意します。

1 ［標準］ツールバーの［メニューエディタ］ をクリックする

2 ［メニューエディタ］ダイアログボックスで、次のメニューを設定する

キャプション (Caption)	名前 (Name)	表示 (Visible)
ファイル	IDMFile	チェック
…開く	IDMOpen	なし
…保存	IDMSave	なし
…-	FileSeparator	なし
…終了	IDMExit	チェック
表示	IDMView	チェック
…フォームの表示	IDMForm	チェック

メニューの設定

メニューを設定する方法については、Lesson13を参照してください。

は「メモ」
は「キーワード」
を表しています

②コードの記述

コードの記述

Form1:Form-MouseDown

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, _
                           X As Single, Y As Single)
    If Button = 2 Then
        FileSeparator.Visible = True
        IDMOpen.Visible = True
        IDMSave.Visible = True
        PopupMenu IDMFile
        IDMOpen.Visible = False
        IDMSave.Visible = False
        FileSeparator.Visible = False
    End If
End Sub
```

Form1:IDMExit-Click

```
Private Sub IDMExit_Click()
    End
End Sub
```

Form1:IDMForm-Click

```
Private Sub IDMForm_Click(Index As Integer)
    Form2.Show 1
End Sub
```

Form1:IDMOpen-Click

```
Private Sub IDMOpen_Click()
    MsgBox "メニューが選択されました"
End Sub
```

Form1:IDMSave-Click

```
Private Sub IDMSave_Click()
    MsgBox "メニューが選択されました"
End Sub
```

Form2:Form-MouseDown

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, _
X As Single, Y As Single)
    If Button = 2 Then
        IDMEdit.Enabled = True
        PopupMenu IDMEdit
        IDMEdit.Enabled = False
    End If
End Sub
```

どのイベントプロシージャにコードを記述するのかが一目でわかります

```
Private Sub IDMExit_Click()
    Unload Form2
End Sub
```

ワンポイントアドバイス

コードウィンドウでのショートカット
- ●Ctrl+F2 ……オブジェクトリストボックスの選択
- ●Tab ………インデント、またはプロジェクトリスト間を移動
- ●Ctrl+I ………クイックヒント表示
- ●Ctrl+Shift+I …パラメータヒント表示
- ●Ctrl+J ………プロパティ／メソッドの一覧
- ●Ctrl+Shift+J …定数の一覧

③コードの解説

解説

1　このプログラムは、ちょっと複雑な動きをします。

①プログラム起動時は、Form1だけが表示されます。このフォームには、［ファイル］と［表示］の2つのメニューが組み込まれています。

Form1
ファイル　表示
フォームの表示

②Form1の上でマウスの右ボタンを押すと、メニューがポップアップします。このメニューには、フォームの［ファイル］メニューに加えて、［開く］と［保存］メニューが追加されています。

Form1
ファイル　表示
開く
保存
終了

③［表示］メニューから［フォームの表示］を選ぶと、もう1つのフォーム「Form2」が表示されます。このフォームは「モーダル」(P.133参照）で起動するため、このフォームを閉じない限りForm1の操作ができま

④このForm2には、［ファイル］メニューだけが表示されています。しかし、フォームの上でマウスの右ボタンをクリックすると、［コピー］メニューがポップアップされます。

Form2
ファイル
コピー

2　フォームのメニューとポップアップメニューは、すべてのメニューをあらかじめメニューエディタで作成しておきます。

ただし、最初から表示しておくメニューと表示しないメニューを、メニューエディタの［表示］チェックボックスで設定しておきます。ポップアップメニューに使うメニューは、［表示］チェックボックスのチェックをはずしておきます。こうしておくと、プログラム実行時は［表示］チェックボックスにチェックを付けたメニューのみが表示されます。

3　では、まずプログラム起動時に表示されるForm1から説明をしていきましょう。

Form1のポップアップメニューは、MouseDownイベントプロシージャを使います。最初に、Ifステートメントを使ってボタンの種類を把握し、右ボタンが押されたときのみ、処理を実行するようにします。このときに行う処理は、プログラム作成時にメニューエディタで［表示］のチェックをはずした、3つのメニューコントロールを［表示］に設定することです。［表示］のプロパティは［Visible］ですから、各コントロールの［Visible］プロパティの値を「True」に設定すると、これらのメニューコントロールは表示状態になります。

```
If Button = 2 Then
    FileSeparator.Visible = True
    IDMOpen.Visible = True
    IDMSave.Visible = True
```

4　メニューを表示状態にしたら、PopUpMenuメソッドを使ってメニューを表示します。このとき、最初の階層のメニューコントロール名（オブジェクト名）を指定すると、次の層のメニューが表示されます。

```
PopupMenu IDMFile, vbPopupMenuRightButton
```

コードの解説も図版を豊富にして1行ずつ解説しています

④まとめと練習問題

11　［編集］メニューの使用状態を有効にしたら、PopupMenuメソッドを使って［編集］メニューを表示させます。メニューがポップアップしたら、再び［編集］メニューを使用不可状態に設定します。

これで、フォームのメニューにないメニュー項目を、ポップアップメニューで表示することができます。

```
PopupMenu IDMEdit, vbPopupMenuRightButton
IDMEdit.Enabled = False
```

Form2
ファイル
終了
コピー

12　メニューの表示が完成したら、メ
します。

［コピー］メニューが選択された
めています。

```
Private Sub IDMCopy_Click
    MsgBox "メニューが選択さ
End Sub
```

13　Form2の［終了］メニューは、Form2自身を終了させて、操作をForm1に戻す必要があります。ですからEndステートメントでプログラムを終了させるのではなく、Unloadステートメントでフォームをメモリから削除（解放）することで、Form2を終了させます。

プログラム自身の終了は、Form2を閉じたあとで、Form1の［終了］メニューで行います。

```
Private Sub IDMExit_Click()
    Unload Form2
End Sub
```

［Unloadステートメントの書式］

```
Unload object
```

object　　アンロードするオブジェクト名またはコントロール名

まとめ

メニューの設定・操作は、あくまでもユーザーが使いやすい形で作成するのが、一番でしょう。わかりやすいユーザーインターフェイスと操作性は、プログラムの質を決めるといっても過言ではありません。

また、Showメソッドを使ったフォームの表示は、ダイアログボックスの表示でよく使うテクニックです。

練習問題

Q1　「Lesson17.vbp」のプログラムのポップアップメニューに、Form2の表示用メニュー項目「このプログラムについて」を追加して、Form2を表示するようにしてください。

Q2　Form2を、イメージコントロールやラベルコントロールを使って、きれいなダイアログボックスに仕上げてください。

また、メニューはすべて削除し、［OK］ボタンを作成してこのボタンでForm2を閉じるようにしてください。

Visual Basic 6.0
サンプル プログラム
OK

解答は巻末に→

ワンポイントアドバイス

PopupMenuで表示できるのは、1つのトップレベルメニューだけです。ですから、いろいろなメニューを組み込みたい場合は、Form2の［編集］メニューのように、ポップアップメニュー専用のメニューを作成しておき、フォームでは非表示にしておいて、ポップアップ時に表示させるようにするといいでしょう。

まとめや練習問題があるので、レッスンの最後に知識を整理することができます

レッスン前のチェックポイント

Visual BasicでWindowsのプログラミングを行うには、Windowsについての基礎知識が必要です。次の8つのポイントで、自分の知識をチェックしてみてください。

POINT

- ❏ファイルのコピーや削除、移動、ファイル名の変更ができますか？
- ❏フォルダの作成や削除、移動ができますか？
- ❏フォルダの階層構造やパス名の意味を理解できますか？
- ❏アプリケーションのメニューやツールバーを知っていますか？
- ❏右ボタンによるショートカットメニューの操作ができますか？
- ❏アプリケーションのインストールやセットアップができますか？
- ❏ウィンドウの各部の名称（ウィンドウ枠、スクロールバー、ステータスバー、ツールバーなど）の機能や役割を理解していますか？
- ❏Visual Basic 6.0を持っている、あるいは購入する予定ですか？

●すべて「YES」 →すぐにでもVisual Basicを学びはじめることができます。レッスンに進んでください！

●「NO」が1～3 →本文中で必要な部分はレッスン中で随時解説しているので大丈夫です。レッスンに進みましょう。

●「NO」が4つ以上 →Visual Basicの前にWindowsの基本操作をマスターしましょう。

CD-ROMの使い方

CD-ROMには完成したプロジェクトファイルを各レッスンごとに収録しました。もしレッスンどおりに入力してもうまく動作しないときは、この「完成版」をチェックして同じように作ってみてください。プロジェクトファイルで使用しているアイコンやイメージファイルも各レッスンフォルダにいっしょに収録しています。必要なファイルをハードディスクにコピーして使ってください。CD-ROMの詳しい使用環境やご注意についてはオリエンテーションの最後のページにある「CD-ROMをご使用になる前に」をお読みください。

Visual Basicの開発環境

Visual Basicにはビジュアルプログラミング環境を強力にサポートする様々な機能が備わっています。

Visual Basicの開発環境

1 作りながら動作できるインタプリタ機能

Visual Basicでは、プログラムを作成したら、すぐにその場でVisual Basic上でプログラムを動作させることができます。

コード全体をすべて書かなくても、部分的な所だけをいったん動作させて、うまくできているかどうかを確認することが可能です。

2 高速なコンパイラ機能

Visual Basicでは、作成したプログラムを、単独実行できる「EXE」ファイルに変換できます。作成したプログラムをVisual Basic上で動作確認をして、最後にこの実行可能なファイルとして作成すれば、Visual Basicがなくても、通常のアプリケーションとして動作させることができます。このファイル形式なら、他のコンピュータで動作させたり、パソコン通信やインターネットを通じて配布することも簡単にできます。

EXEファイルの作成

実行可能ファイルのオプション

コンパイルオプション

3 ユーザーインターフェイスの作成が簡単

Visual Basicのプログラミングは、「コントロール」と呼ばれるWindowsの部品を組み合わせて、プログラムの外観（ユーザーインターフェイス）を作成します。

ユーザーインターフェイスの作成はとても簡単で、ツールボックスに配置されているコントロールを、「フォーム」と呼ばれるウィンドウ枠の中にマウスでドラッグするだけでできてしまいます。

ツールボックスからコントロールを自由に配置

4 コード作成用のエディタを標準装備

Visual Basicでは、コントロールを使ってユーザーインターフェイスを作成しながら、同時に実際の処理を受け持つ「プログラムコード」を記述します。コードの作成は、専用のエディタ（編集ソフト）が組み込まれており、色によるキーワードの識別や入力候補を表示するなどの、数々の入力支援機能を持っています。

```
Dim ImgNo As Integer

Private Sub ExitBtn_Click()
    End
End Sub

Private Sub Form_Load()
    ImgNo = 1
    Timer1.Interval = 100
    StartBtn.Enabled = False

End Sub


Private Sub IDMExit_Click()
    End
End Sub
```

コードエディタ

5 プロジェクト方式を採用

これまでのBasicプログラミングでは、1つのプログラムに対して1つのソースコードで作成されていました。これに対して、Visual Basicのプログラミングでは、「プロジェクトファイル」と呼ばれるファイルで、プログラムのソース（元になるコードファイル）を管理します。これにより、複数のファイルを組み合わせたプログラム開発が可能になり、以前使ったフォームやコードを、別のプログラムで再利用することが可能になっています。

複数のファイルをプロジェクトで扱うことが可能

6 モジュールプログラミングが可能

プロジェクト方式を採用したことで、ソースを複数のメンバーで分割して作成することが可能になりました。また、プログラミング言語自体が「構造化プログラミング」の概念を取り入れているので、プログラムをモジュールごとに組み上げることができます。

例えば、1つのプログラムを「メニューモジュール」「設定モジュール」「ファイル保存モジュール」などと、機能ごとに分割して組み立てが可能です。これにより、開発が容易に行えると同時に、デバッグ作業の軽減にもつながっています。

モジュールの追加

フォームモジュール

標準モジュール

7 APIいらずの豊富なメソッド・関数（Professional Edition以上）

Windowsのプログラミングは、本来「API」と呼ばれるWindowsの機能を呼び出して組み立てていくものです。しかし、このAPIはC言語などから使うようにできており、その記述も複雑です。

Visual Basicでは、このAPIの多くをコントロールの「メソッド」や「関数」に組み込んでおり、APIよりも簡単な記述で同じような機能が使えるようになっています。このおかげで、Visual Basicを使ったWindowのアプリケーション開発がわかりやすく容易なものになっています。

APIビューア

8 デバッグ機能を装備

Visual Basicでは、変数や式の中身を調べる「ウォッチ」や、プロシージャのテストや式の評価をするときに便利な「イミディエイト」、プログラムを一行ずつ実行する「ステップ実行」機能など、プログラムの不具合（バグ）を調べ、修正する機能を装備しています。いずれも、簡単な設定で操作でき、デバッグ作業の負担を軽減できます。

デバッグ機能を標準装備

9 ディストリビューションウィザードを用意

質問に答えていくだけで、作成したプログラムを配布する用意ができる「ディストリビューションウィザード」が、Visual Studio 6.0のツールとしてインストールされています。

Visual Basicで作成したプログラムを配布するには、関連する「DLL」ファイルをいっしょに配布しないと、正常に動作しません。ディストリビューションウィザードは、プロジェクトファイルからこれらを自動解析し、必要な関連ファイルを実行ファイルといっしょに圧縮してまとめてくれます。また、インストールするための「セットアッププログラム」も自動で組み込んでくれますので、プログラム配布キットを簡単に作成できます。

ディストリビューションウィザード

10 強力なオンラインヘルプ機能

これまで、Windowsのヘルプシステムで提供されていた、Visual Basicのオンラインヘルプは、新しくHTMLをベースとしたヘルプシステムに変更されました。

また、提供されるヘルプコンテンツも、「MSDN※ライブラリ」という形態で、Visual Studio 6.0のヘルプとして参照できるようになっています。この中には、Visual Basicのランゲージリファレンスだけでなく、Microsoftが提供している開発キットのマニュアルも含まれ、開発者にとっては貴重な情報源となっています。

このオンラインヘルプを使えば、Visual Basicのマニュアルに書かれていることをすべてオンラインで参照することができます。プログラミング中に不明な用語や事柄が見つかった場合、すぐに検索でき、画面上で参照したり、印刷することができます。

※MSDN
Microsoft Developer Networkの略で、Microsoftが有償で提供している、開発者向けの技術情報サービス。市販では手に入れることのできない技術情報や、ソフトウェア開発キットなどが供給される。

MSDNライブラリで提供される
Visual Basic オンラインヘルプ

11 DHTML作成機能（Professional Edition以上）

Visual Basic 6.0から追加された新しい機能で、DHTML（Dynamic HTML）を使ったHTMLファイルを作成することができる機能です。

この機能を使うと、HTMLページをフォームのように使用して、コントロールを配置し、コードを使った処理を組み込むことができます。作成したDHTMLプロジェクトをコンパイルすると、「DLL」ファイルとして生成されます。このDLLファイルを、作成したHTMLファイルと同じ場所に配置すれば、ブラウザで機能を実行させることができます。

Visual Basicの開発環境から作成できるので、DHTMLをWindowsプログラムと同じ操作手順を活用できます。

DHTMLプロジェクト作成画面。HTMLのレイアウトもいっしょにできる

インターネットエクスプローラ4.01で実行

Visual Basic の画面構成

Visual Basic の各部分の名称を説明します。

Visual Basicの各部の名称

④フォーム作成ウィンドウ
⑫オブジェクトブラウザ
⑥プロジェクトエクスプローラ
②ツールバー
⑤コードウィンドウ
①メニュー
⑭カラーパレット
⑦プロパティウィンドウ
⑨イミディエイトウィンドウ
⑬データビューウィンドウ
⑩ウォッチウィンドウ
⑪ローカルウィンドウ
③コントロールツールボックス
⑧フォームレイアウトウィンドウ

①メニュー

Visual Basicの操作を行うメニューです。モジュールの挿入や削除なども、このメニューから実行できます。

②ツールバー

メニューの中でよく使う項目が、ツールバーのボタンになっています。ツールバーの上でマウスの右ボタンを押すと、表示するボタンの種類を設定できます。

③コントロールツールボックス

コントロールを配置してあるツールボックスです。ここから、使用したいコントロールを選んで、フォームに配置します。

④フォーム作成ウィンドウ
プログラムのベースとなるウィンドウを「フォーム」と呼びます。このフォーム作成ウィンドウで、コントロールをフォームに配置し、プログラムのユーザーインターフェイスを作成します。

⑤コードウィンドウ
プログラムの処理を実行する、コードを作成するウィンドウです。テキストエディタ機能を持ち、コードのコピーや貼り付けなども実行できます。

⑥プロジェクトエクスプローラ
プロジェクトファイルを管理するウィンドウです。操作したいフォームやコードウィンドウの切り替えを行います。

⑦プロパティウィンドウ
フォームやコントロールが持つ様々な属性情報「プロパティ」を設定するウィンドウです。フォーム作成ウィンドウで、コントロールやフォームをクリックすると、自動的にこのプロパティウィンドウもそのオブジェクトのものに切り替わります。ここで操作できるプロパティは、デザイン時に値の設定ができるものが表示されます。

⑧フォームレイアウトウィンドウ
プログラム実行時に、Windowsの画面の中でフォームの表示される位置を設定するウィンドウです。

⑨イミディエイトウィンドウ
プログラムのコード作成中に、一時的にコードを実行して動作を確かめたい場合、このウィンドウに記述します。

⑩ウォッチウィンドウ
プログラム動作中に、式や変数の値を調べることができます。調査中の値は、このウィンドウに表示されます。

⑪ローカルウィンドウ
指定したプロシージャ内の、すべての変数とその値を表示するウィンドウです。

⑫オブジェクトブラウザ
プロジェクトで使用できる、すべてのオブジェクトと関連するメソッド・プロパティを見ることができるブラウザです。検索機能を持ち、このウィンドウで使用したいオブジェクトやプロパティを、クリップボード経由でコードウィンドウに貼り付けることができます。

⑬データビューウィンドウ（Professional Edition以上）
データビューウィンドウは、プロジェクトに追加されたすべての「Data Environment」を確認して、「Data Environment」との接続状態を表示します。
また、データベースのテーブルやビューの作成や修正、「ストアドプロシージャ」および「トリガ」と呼ばれる特殊なストアドプロシージャの作成修正などを行うことができます。

⑭カラーパレット
フォームやコントロールの色を選択・設定するパレットです。
プロパティウィンドウで色を設定する場合にも、カラーパレットは表示されますが、それ以外に専用のウィンドウで表示できます。

Visual Basicの基本操作

Visual Basicの起動と終了、ファイルの保存について説明します。

1 Visual Basicは、インストールするとスタートメニューに登録されています。Visual Basicを起動するときは、Windowsの［スタート］メニューで、［プログラム］→［Microsoft Visual Basic 6.0］→［Visual Basic 6.0］の順にクリックします。

1 ここをクリック

スタートメニューに登録されている

2 起動すると、作成するプログラムの種類を選ぶウィンドウが表示されます。通常のプログラム作成であれば、「標準EXE」を選び、［開く］をクリックします。

1 ここをクリック

この画面はVisual BasicのEditionによって異なります。

2 ここをクリック

フォームが用意される

3 すでに作成したファイルを使用する場合は、手順2の画面で［既存のファイル］または［最近使ったファイル］タブをクリックして選びます。

1 ファイルを選択
2 ここをクリック

4 プログラムを作り終えたらファイルを保存します。作成したプログラムを保存するときは、プロジェクトファイルと、プロジェクトに含まれるフォームやモジュールファイルを保存します。

1 ここをクリック

5 Visual Basicを終了するときは、［ファイル］メニューの［Microsoft Visual Basicの終了］を選びます。未保存のファイルがある場合は、保存するようにメッセージが表示されます。

保存を確認するメッセージ
1 ここをクリック

CD-ROMをご使用になる前に

CD-ROMを使用する際のご注意等です。CD-ROM中のテキストファイル「readme.txt」もあわせてお読みください。

●CD-ROMについて

巻末に付属のCD-ROMには本書「10日でおぼえるVisual Basic 6.0入門教室」(VBテックラボ＆瀬戸遙著)の本文中で解説した各レッスンの完成版やレッスンに必要なファイルが入っています。ハードディスクにフォルダごとドラッグコピーしてお使いください。ファイルをコピーする方法がわからないときはWindowsのオンラインヘルプから「ファイルやフォルダをコピーするには」をご覧ください。

※ レッスンで使用するプロジェクトファイル(またはHTMLファイル)で、外部ファイルとリンクしているものはリンクの参照情報(フルパス名)が無効になっている場合があります(特にLesson6/23/29/37/38など)。その際は、必要なファイルをハードディスクにコピーした後、本書の手順に従い自分の環境に合うように再度プロジェクトファイル等を作りなおしてください。また、Lesson29で使用するデータベースファイルについては、コピーした後[属性]の[読み取り専用]のチェックをはずす必要があります。

●免責事項について

CD-ROMに収録されたファイルは通常の運用においては何ら問題ないことを編集部では確認していますが、万一運用の結果、いかなる損害が発生したとしても著者及び(株)翔泳社はいかなる責任も負いません。すべて自己責任においてご使用ください。

●CD-ROMの使用環境条件

CD-ROMにVisual Basic 6.0は付属していません。別途購入する必要があります。その他、全レッスンを進めるうえで以下のソフトやハードを用意しておくと便利です。

・サウンドファイルやMIDIファイルを再生するためのオーディオカード
・Internet Explorer 4.01およびFrontpage Express(Microsoft社より無償提供。Windows98を使用している方は不要)

●CD-ROMのテスト環境について

CD-ROMは以下の環境で正常に動作することを確認しました(Learningは部分確認)。

・Visual Studio 6.0 Enterprise Edition/Visual Basic 6.0 Learning Edition
・Windows98/Windows NT Workstation 4.0
・Compaq Deskpro 4000 5166 M2500 CDS/RAM 128MB/HD 2.1GB

●著作権等について

本書に収録したプログラムおよびソースコードの著作権は、著者および(株)翔泳社が所有します。ただし読者が個人的に利用する場合に関してはソースコードやフォーム、コントロールの流用や改変は自由です。商用利用に関しては、(株)翔泳社へ電子メール(mail-web@shoeisha.co.jp)などでご一報ください。

(株)翔泳社 編集部

Let's VBSchool!!

第1日 (Lesson1 ～ Lesson7)

はじめてのプログラムを作る

第1日は、まずVisual Basicの各ツールを使ってプログラムの外観を作成しましょう。また、プログラム作成の流れも実際に操作しながら覚えます。

プログラムのユーザーインターフェイスを作成するには、フォームにコントロールと呼ばれる様々な部品を配置して、その外観を整えます。

コントロールには、ラベルコントロールやイメージコントロールのように、ユーザーへ情報を表示するために使用するコントロールと、コマンドボタンやリストボックスのように、ユーザーの操作に対応して動作するコントロールの2種類があります。中でも比較的使用方法が簡単なコントロールをいくつか使って、コントロールをフォームに配置する方法や、コントロールの属性を決めるプロパティの操作方法を紹介します。

今日のレッスンで解説する項目

- ❏ プログラム作成の流れの把握
- ❏ Visual Basicの各ツールの操作方法
- ❏ 複数のコマンドボタンコントロールを使う際のプロシージャの使い分け
- ❏ ラベルコントロールの使い方
- ❏ コントロールのプロパティを変更する方法
- ❏ コードでコントロールのプロパティを変更する方法
- ❏ イメージコントロールに画像ファイルを設定する方法
- ❏ イベントプロシージャを使ったテクニック
- ❏ コントロールを使ってフォームをデザインする方法

Lesson 1

第１日
１時限目

はじめての
プログラムを
作る

Lesson1.vbp
VB6CD
Lesson01

◎Learning Edition

コマンドボタンでメッセージを表示する

ひととおり、プログラム作成の流れを把握します。また、Visual Basicの各ツールの操作方法を覚えます。

● 今回作成するプログラム

Form1
押す

1 このボタンを押すと…

Form1
lesson1
はじめてのプログラムです。
OK
押す

2 メッセージが表示される

3 [OK] ボタンでメッセージが閉じる

POINT

では、はじめてのプログラムを作りましょう。このプログラムは、コマンドボタンを押すと、メッセージボックスを表示するだけのシンプルなプログラムです。このプログラムの作成を通じて、コントロールのフォームへの配置方法とプロパティの変更方法、プログラムコードの記述方法、プログラムの実行方法を学習します。

コントロールとプロパティの設定

1 [スタート] メニューからVisual Basicを起動する

1 ここをクリック

2 [新しいプロジェクト] ダイアログボックスから [標準 EXE] を選択して [開く] をクリックする

1 [標準.EXE] を選択

2 [開く] をクリック

右欄参照

この画面はEditionによって異なる

[スタート] メニュー

アプリケーションなどを起動するときに使用するWindowsのタスクバーのメニューです。画面の左下にあります。
(VisualBasicの起動と終了はP.xxxを参照)

Visual Basicを起動するには

[スタート] ボタン、[プログラム]、[Microsoft Visual Basic 6.0]、[Microsoft Visual Basic 6.0] の順にクリックします。

標準.EXE

「EXE」とは実行ファイルのことです。一般的なアプリケーションを作るときはこの「標準.EXE」を選択します。

次回からこのダイアログボックスを表示しない

[次回からこのダイアログボックスを表示しない] チェックボックスをクリックしてチェックしておくと、次回からこのダイアログボックスが表示されなくなり、新しいプロジェクトとフォームが自動的に作成されます。

Visual Basicの初期画面

これがVisual Bassicの基本となる画面です。通常はここからいろいろなプログラムを作っていきます。

オブジェクト

オブジェクトには様々な意味がありますが、フォームやコントロールなど、プログラムを構成する部品を総称して「オブジェクト」と呼んでいます。

●代表的なオブジェクト

・コントロールすべて
・Formオブジェクト
・Appオブジェクト
・データアクセスオブジェクト
・Clipboardオブジェクト
・Databaseオブジェクト
・Fontオブジェクト
・HyperLinkオブジェクト
・Lightオブジェクト

3 新しいプロジェクトとフォームが作成される

プロジェクトエクスプローラ

プロパティウィンドウ

フォームウィンドウ

プロジェクトウィンドウ

COLUMN

プロジェクトついて

これまでのBasicプログラムは、1つのソースファイルで1つのプログラムを実行していました。

Visual Basicで作成するプログラムは、複数のファイルで構成されます。フォームを1つ使うと、プロジェクトウィンドウにフォームのファイルが追加されます。標準モジュールを追加すれば、このファイルがプロジェクトに追加されます。このように、1つのプログラムを作成するのに複数のファイルが必要になるため、どのプログラムはどのファイルを使うのかを管理する仕組みが必要になってきます。これが、「プロジェクト」という管理方法です。

プロジェクトは「プロジェクトファイル」という専用のファイルを持ち、ここにプログラムに使うファイルのリストを記述しています。

プロジェクトファイル

ファイルの中身

COLUMN ……

フォームについて

プログラムのベースとなるウィンドウのことです。ここに、コマンドボタンやメニューなどのコントロールを配置して、プログラムの外観を作成します。

1つのプログラムで複数のフォームを持つことができます。

4 ツールボックスの［CommandButton］をクリックする

Project1 - Microsoft Visual Basic [デザイン]

1 このボタンをクリック

5 ドラッグして適当な大きさの四角形を描きボタンを作成する

コントロール類はドラッグして配置する

1 ここをクリックして…

2 そのままここまでドラッグ

2415 × 855

ボタン名の表示

ツールボックスやツールバーのボタンの上にマウスポインタを移動すると、そのボタンの名前がポップアップで表示されます。

ボタンを作成するには［CommandButton］をクリックし、フォーム上でドラッグします。

ドラッグ

マウスのボタンを押したまま、マウスポインタを別の位置まで移動することです。ほとんどのコントロールは、ドラッグしてフォームの上に配置していきます。

マウスポインタの形

フォームウィンドウにマウスポインタを移動すると、ポインタの形が十字形に変わります。

COLUMN ……

コントロールは「部品」

プログラムを形作る「部品」をコントロールといいます。コントロールには、ラベルやフレーム、イメージコントロールなど文字や画像を表示するタイプのものと、コマンドボタンやリストボックスのように、ユーザーによって動作するタイプのものがあります。これらのコントロールをフォームに配置して、プログラムの外観やユーザーインターフェイスを作成します。

6 作成されたボタンをクリックして選択状態にする

Form1

Command1

1 ボタンをクリック

7 プロパティウィンドウで［Caption］プロパティの「Command1」を「押す」に変更する

プロパティ - Command1

Command1 CommandButton

全体 | 項目別

(オブジェクト名)	Command1
Appearance	1 - 3D
BackColor	&H8000000F&
Cancel	False
Caption	押す
CausesValidation	True
Default	False
DisabledPicture	(なし)
DownPicture	(なし)
DragIcon	(なし)
DragMode	0 - 手動
Enabled	True
Font	MS Pゴシック
Height	735

オブジェクトボックス

タブ

1 ここをダブルクリック

2 「押す」を入力

プロパティリスト

オブジェクトの選択とハンドル

オブジェクトが選択された状態になると、その周りに黒い四角形が表示されます。この四角形は、サイズ変更ハンドルと呼ばれます。

プロパティはどこにある？

プロパティウィンドウは通常画面の右端にあります。(P.4参照)

「Command1」を変更するには

［Caption］という項目名をダブルクリックして、「Command1」を反転表示（選択）し、キーボードから「押す」と入力します。

プロパティとは？

プロパティについては、Lesson3のコラムで説明しています。(P.22参照)

8 ボタンの名前が「押す」になった

9 ボタンを下に移動し、ダブルクリックする

変更が即座に反映される

VisualBasicでは、ボタンなどのオブジェクトに対するプロパティの変更内容はすぐにフォームに反映されます。

ボタンの移動中は…

ボタンをドラッグすると、移動先のボタンの位置を示す白い四角形が表示されます。

10 表示されたコードウィンドウにコードを書く

```
Private Sub Command1_Click()
    MsgBox "はじめてのプログラムです。"
End Sub
```

1 ここにコードを書く

プログラムの実行と終了

1 ［開始］ボタンをクリックする

1 このボタンをクリック

COLUMN

プログラムの実行ーインタプリタ機能

Visual Basicの［開始］ボタンをクリックして、作成したプログラムを実行できますが、これはVisual Basicの上でプログラムを実行させています。この時点では、Visual Basicを起動しないと、プログラムは実行できません。この機能を「インタプリタ」機能といいます。これに対し、Visual Basicがなくてもプログラムを実行できるようにするには、「単独で実行可能なEXEファイルの作成」を行う必要があります（詳しくはP.338参照）。

コードウィンドウ

プログラムのコードを入力するためのウィンドウです。「コードエディタウィンドウ」または「コードウィンドウ」と呼ばれています。

コードの意味については、P.10以降のコードの「解説」を参照してください。

プログラムの実行について

ここでは、Lesson1なのでプログラムの実行と終了方法について説明します。この後のレッスンでは説明しませんので、この手順を参考にして実行と終了を行ってください。

その他の実行方法は？

実行メニューから［開始］を選択してもプログラムを実行することができます。ショートカットはF5キーです。

2 フォームが起動した後［押す］ボタンをクリックする

Form1

押す

1 このボタンをクリック

3 メッセージボックスが表示されるので［OK］ボタンをクリックする

Form1

lesson1

はじめてのプログラムです。

OK

押す

1 このボタンをクリックしてメッセージボックスを閉じる

4 ［閉じる］ボタンをクリックしてフォームを終了する

Form1

押す

1 このボタンをクリック

フォームが起動する場所を変更するには？

画面右下にある［フォームレイアウト］のモニタ中のウィンドウアイコンを好きな位置に移動することで、フォームが起動する場所を変えることができます。

その他の終了方法は？

ツールバーの［終了］ボタンをクリックしても、フォームを終了することができます。

解 説

1 フォームに配置したコマンドボタンをマウスでダブルクリックすると、コードウィンドウが開き、コマンドボタンが押されたときに実行されるプロシージャが表示されます。

Form1

押す

1 ボタンをダブルクリックすると…

コードウィンドウが表示される

Project1 - Form1 (コード)

Command1 / Click

```
Private Sub Command1_Click()

End Sub
```

2 ここにコードを書く

COLUMN

プロシージャについて

プログラムのコードを、小さなまとまり（ブロック）ごとに区切ったものです。Visual Basicでは、このプロシージャ単位で処理を組み立てていきます。

プロシージャの作成には、SubステートメントやFunctionステートメントを使って、「ここからプロシージャがはじまります」という宣言を行い、1つのコードブロックを作成します。「End Sub」はSubステートメントの終了を意味します。

イベントが発生すると、自動的に「イベントプロシージャ」と呼ばれる専用のプロシージャが呼び出されます。たとえば、「Command1」というコマンドボタンをクリックすると「Clickイベント」が発生し「Command1_Clickイベントプロシージャ」が呼び出されます。

このように、各オブジェクトごとに対応したイベントが設定されており、オブジェクト名とイベント名を組み合わせたプロシージャ名が使われます。これについては、Lesson8でも解説しているので参照してください。

2 コマンドボタンがクリックされたときに、なにか処理をしたい場合は、この「Private Sub Command1_Click()」プロシージャ内にコードを記述します。

```
Project1 - Form1 (ｺｰﾄﾞ)
Command1                Click
Private Sub Command1_Click()
|
End Sub
```

【イベントプロシージャの宣言部分】

```
Private Sub Command1_Click()
  ①      ②     ③       ④
```

①他のプロシージャからのみ参照できるプロシージャ「Private」の指定。省略可能ですが自動的に入力されているので省略する必要はないでしょう。
②Subステートメント。プロシージャの開始を宣言します。
③オブジェクト名。処理の対象となるオブジェクトの指定です。
④イベント名。処理の対象となるイベント名です。③のオブジェクトと「_」(アンダースコア) でつないで記述します。

3 メッセージボックスを表示する機能は、MsgBox関数が行います。関数名「MsgBox」の後に、表示したい文字を"(半角引用符) で囲んで記述します。

【MsgBox関数の書式】

```
MsgBox message
```

message　　表示するメッセージを記述します。文字列は引用符で囲みます。

COLUMN ・・・・・

関数とは

関数は、ある値を指定して実行すると、実行結果を値で返してきます。たとえば、Sin関数は、ある角度を与えて実行すると、-1から1の間の値を返してきます。また、FileLen関数は、ファイル名を指定すると、そのファイルのサイズを返してきます（関数によっては、引数を持たないものや、値を返さないものがあります)。

Visual Basicでは、さまざまな関数が用意されており、行いたいプログラムの処理に合わせた関数を使ってプログラムを作成していきます。

MsgBox関数は、実際には多くのオプション設定ができるのですが、メッセージを表示するだけであれば、この書式で実行できます（正確な書式は巻末の関数リファレンスを参照)。P.192～P.194では、さらにMsgBox関数の詳しい使い方を解説しています。

「MsgBox」と入力した時点で「パラメータヒント」が表示されます。これは、MsgBox関数で入力可能なパラメータを即座に表示する機能です。MsgBox関数の場合はこのように表示されます。パラメータヒントをすぐに消したいときは、Escキーを押します。

Project1 - Form1 (ｺｰﾄﾞ)
Command1 | Click

```
Private Sub Command1_Click()
    MsgBox |
End  MsgBox(Prompt, [Buttons As VbMsgBoxStyle = vbOKOnly], [Title], [HelpFile], [Context]) As VbMsgBoxResult
```

パラメータヒント

```
MsgBox(Prompt, [Button As VbMsgBoxStyle = vbOkOnly],
[Title], [HelpFile], [Context]) As VbMsgBoxResult
```

4 MsgBoxに続けて「"はじめてのプログラムです。"」と入力します。コードはこれで完成です。右上の［閉じる］ボタンをクリックしてコードウィンドウを閉じておきましょう。

Project1 - Form1 (ｺｰﾄﾞ)
Command1 | Click

```
Private Sub Command1_Click()
    MsgBox "はじめてのプログラムです。"
End Sub
```

このボタンをクリック

COLUMN ・・・・・

コードウィンドウについて

コードウィンドウを呼び出すには、①F7キーを押す、②［表示］メニューから［コード］を選択する、③配置されたオブジェクトをダブルクリックする、の3種類があります。

コードウィンドウの上にある▼をクリックすると、項目がリスト表示され、オブジェクトやイベントを選択することができます。左にはコードを記述可能なオブジェクトの一覧があり、右にはイベント（プロシージャ）一覧があります。コードが記述されているイベントプロシージャは、太字表示になっています。

(General)
Command1
Form

Click
DragDrop
DragOver
GotFocus
KeyDown
KeyPress
KeyUp
LostFocus
MouseDown
MouseMove
MouseUp
OLECompleteDrag

太字になっている

オブジェクト一覧

プロシージャ一覧

まとめ

Visual Basic でプログラムを作成する場合は、
①フォームとコントロールでユーザーインターフェイスを作成する
②フォームやコントロールのプロパティを変更し、外観を整える
③プログラムの動作をコードで記述する
という流れになります。

フォームやコントロールは、あくまでも見た目やユーザーの操作を受け付ける外観を作成するものであり、各コントロールの実際の処理は、コードを記述して作成します。

練習問題

Q1　メッセージボックスのメッセージを「こんにちは、みなさん！」に変えて表示するようにしてください。

解答は巻末に→

ワンポイントアドバイス

コントロールをフォームに配置するには、ツールボックスのコントロールアイコンをクリックし、フォームの上でマウスの左ボタンを押しながらドラッグします。適当なサイズになったところでボタンを離すと、その大きさで確定します。

Form1　3495 x 375　Form1　Command1

1　ドラッグする

あるいはツールボックスのコントロールアイコンをダブルクリックすると、標準的な大きさのコントロールがフォームの中心に配置されます。

Lesson 2

第１日
１時限目

はじめての
プログラムを
作る

Lesson2.vbp
VB6CD
Lesson02

◎Learning Edition

終了ボタンを追加する

複数のコマンドボタンコントロールを使う際のプロシージャの使い分けを覚えます。

● 今回作成するプログラム

Form1
押す
終了

このボタンは
Lesson1と同じ

このボタンを押すと…

プログラムが終了して
ダイアログボックスが閉じる

POINT

最初に作成したプログラムに、プログラムを終了するためのボタンを付けます。ボタンは、コマンドボタンコントロールを使用し、ここにプログラムを終了させるステートメント「End」を記述します。

ステートメントとは、プログラムの流れを制御したり、コードを書きやすくするなど、プログラム作成を行う上で必要な命令のことです。Visual Basicのプログラム作成は、これらのステートメントを組み合わせて、オブジェクトのプロパティやメソッドを操作して行います。

ここでは、コマンドボタンコントロールのサイズの変更や、ボタン表面の文字を変更する方法を覚えます。また、コマンドボタンが押されたときの処理（イベントプロシージャ）の作成方法を覚えます。

コントロールとプロパティの設定

1 [押す] ボタンを上に移動する

Form1

押す

1 ここまでドラッグ

2 ツールボックスの [CommandButton] をクリックする

標準

Project1 - Form1 (Form)

Form1

CommandButton

押す

1 このボタンをクリック

3 フォーム上でドラッグし、コマンドボタンコントロールを1つ作成する

Form1

押す

1815 x 615

1 ここをクリックして…

2 そのままここまでドラッグ

使用するプロジェクト

Lesson1で作成したプロジェクトを引き続き使用してください。

または、CD-ROMの「Lesson01」フォルダにある「Lesson1.vbp」を使用してください。

ボタンの作り方

ボタンの作り方はLesson1と同じです。

値の変更方法

プロパティの値を変更する方法については、Lesson1を参照してください。

ダブルクリックでの編集

プロパティウィンドウでプロパティを変更するとき、オブジェクト名や値の部分をダブルクリックすることで値が選択状態になり、すぐに編集にとりかかれます。もちろんクリックでもかまいません。クリックのときは値の部分をクリックします。

コードウィンドウの開き方

コードウィンドウの開き方は、Lesson1と同じです。

4 作成したボタンを選択し、プロパティウィンドウで[Caption]プロパティの値を「終了」に変更する

(オブジェクト名)	Command2
Appearance	1 - 3D
BackColor	&H8000000F&
Cancel	False
Caption	終了
CausesValidation	True
Default	False
DisabledPicture	(なし)

1 ここをダブルクリック

2 「終了」を入力

5 ボタンの名前が終了になった

6 ボタンをダブルクリックして、コードウィンドウを表示し、コードを記述する

```
Private Sub Command1_Click()
    MsgBox "はじめてのプログラムです。"
End Sub

Private Sub Command2_Click()
    End
End Sub
```

1 ここにコードを書く

プロジェクトの解放

1 プロジェクトを解放するには［ファイル］メニューから［プロジェクトの解放］を選択

1 ここをクリック

2 ここをクリック

2 ［はい］をクリックして変更点を保存する

1 ここをクリック

3 新しいフォルダを作成しフォルダ名を「Lesson02」と名付け、そのフォルダをダブルクリックで開く

1 ここをクリック

2 名前を変更

3 ここをダブルクリック

プロジェクトの解放と保存

プログラムを作り終えたらいったんプロジェクトを解放するか、もしくはファイルを保存しておきましょう。

プロジェクトファイルを閉じることをプログラミング用語で「解放」と呼んでいます。これはプロジェクトが使用していたメモリを他のプロジェクトが使えるように解放するためそう呼ばれています。

プロジェクトが保存されていないときは？

プロジェクトが保存されていないときは、左図のようなダイアログボックスが表示され、作業の変更点を保存するかどうかたずねてきます。

レッスン用フォルダを作る

レッスンごとにプロジェクトを保存するフォルダを作っておき、後で整理しやすくします。

4 「Form1.frm」の名前でフォームを保存した後、続けて「Lesson2.vbp」の名前でプロジェクトを保存

ファイル名(N): Form1.frm
ファイルの種類(T): フォーム モジュール (*.frm)
保存(S) キャンセル ヘルプ(H)

1 ここをクリック

ファイル名(N): Lesson2.vbp
ファイルの種類(T): プロジェクト ファイル (*.vbp)

2 名前を変更

3 ここをクリック

5 プロジェクトが解放された

プロジェクトエクスプローラの表示が空になる

プロジェクトの保存

1 プロジェクトを保存するには［ファイル］メニューから［プロジェクトの上書き保存］を選択する

Project1 - Microsoft Visual Basic [デザイン]
ファイル(F) 編集(E) 表示(V) プロジェクト(P) 書式(O) デバッグ(D) 実行(R) クエリー(U)
新しいプロジェクト(N) Ctrl+N
プロジェクトを開く(O)... Ctrl+O
プロジェクトの追加(D)...
プロジェクトの解放(R)
プロジェクトの上書き保存(V)
名前を付けてプロジェクトの保存(E)...
Form1.frm の上書き保存(S) Ctrl+S
名前を付けて Form1.frm の保存(A)...
選択項目の上書き保存(L)
スクリプトの変更の保存(H)
印刷(P)... Ctrl+P
プリンタの設定(U)...

1 ここをクリック

2 ここをクリック

保存の順序

新しいプロジェクトを保存するときは、フォームの保存→プロジェクトの保存、という順番になります。フォームの拡張子は「.frm」、プロジェクトの拡張子は「.vbp」となっています。

プロジェクト解放後

プロジェクトを解放した後はオブジェクトウィンドウには何も表示されなくなります。

ボタンで保存

［プロジェクトの上書き保存］ボタンを押して保存することもできます。

フォームだけを保存する

プロジェクトを保存せずに、フォームだけを保存することもできます。［ファイル］メニューから［Form1の上書き保存］を選んでください。

プロジェクトを開く

1 保存したプロジェクトを開くには［ファイル］メニューから［プロジェクトを開く］を選択する

1 ここをクリック
2 ここをクリック

2 ダイアログボックスで先ほど保存した「Lesson2.vbp」を選択し、［開く］をクリックする

1 ここをクリック
2 ここをクリック

3 プロジェクトが開く

ボタンで開く

ツールバーの［プロジェクトを開く］ボタンをクリックしても同じ操作ができます。

［最近使用したファイル］

［最近使用したファイル］タブをクリックすると、以前に使用したファイルのリストが表示されます。よく使うファイルをを探すときに便利です。

フォームが表示されないときは？

プロジェクトを開いたとき、何も表示されないときは、プロジェクトウィンドウからフォームを開く必要があります。次ページのコラムを参照してください。

便利なショートカット

オブジェクトウィンドウや、コードウィンドウは、キーボードのショートカットを使って呼び出すことができます。知っていると便利なショートカットを集めてみました。

F7
→コードウィンドウの表示
Shift+F7
→オブジェクトウィンドウの表示
F4
→プロパティウィンドウの選択
Ctrl+Tab
→ウィンドウの切り替え
F5
→プログラムの実行

COLUMN ・・・・・

プロジェクトの閉じるボタン

プロジェクトウィンドウの閉じるボタンをクリックしても、プロジェクトが解放されたことにはなりません。これはフォームオブジェクトが閉じた状態になっただけです。フォームを再度表示するには、プロジェクトエクスプローラからフォームのアイコンをダブルクリックするか、［表示］メニューから［オブジェクト］を選択します。

解説

1　［終了］ボタンをダブルクリックすると、このボタン（Command2）が押されたときのプロシージャ「Private Sub Command2_Click()」が表示されます。ここに、「End」と記述します。Endステートメントは、単独で記述すると、プログラム自身を終了させる命令になります。

```
End
```

まとめ

複数のコマンドボタンを配置したときは、それぞれのボタンのプロシージャに、その処理を記述します。この例題では、Command1に「メッセージを表示」、Command2に「プログラムを終了」と、2つの処理を2つのボタンで実行するように設定しました。配置したコントロールをダブルクリックすると、そのコントロールに対応したプロシージャを呼び出すことができます。

コントロールを配置し終えたプロジェクトファイルは、保存するようにしましょう。特にプログラムをインタプリタ機能で実行するときは、不具合やエラーでシステムがハングアップする可能性もあります。一度保存をしてからプログラムを実行した方がよいでしょう。

フォームやプロジェクトファイルは、別の名前でいくつも保存しておくことができます。ユーザーインターフェイスのいくつかのパターンやバリエーションを保存して、あとから再利用することもできます。

練習問題

Q1　［終了］ボタンの文字を「閉じる」に変更してください。

解答は巻末に→

ワンポイントアドバイス

Visual Basicで作成するプログラムの動作は、コントロールのプロパティを変更したり、関数を使って組み立てていきます。

また、処理の流れを制御したり、コードを書きやすくするなどの、プログラム作成を行う上で必要な命令文を、「ステートメント」と呼んでいます。

Lesson 3

第1日 2時限目

はじめてのプログラムを作る

Lesson3.vbp
VB6CD
Lesson03

◎Learning Edition

プロパティを操作する

ラベルコントロールを使いながら、コントロールのプロパティを変更する方法を覚えます。

● 今回作成するプログラム

Form1
楽しいVisual Basic

フォームにラベルコントロールを追加する

COLUMN

プロパティについて

プロパティとは「属性」のことで、コントロールやフォームの外観・形状や、動作状態を設定したり把握するのに使います。たとえば、フォームの［BackColor］プロパティは、フォームの背景色を設定します。コマンドボタンの［Caption］プロパティは、ボタン表面の文字を設定します。

各コントロールに用意されている［Enabled］プロパティは、コントロールを有効にしたり無効にしたりします。よく、アプリケーションなどを使っていると、コマンドボタンがグレー表示になって押せない状態にあるのを見かけますが、これは［Enabled］プロパティを操作することで実現しています。

プロパティには、それぞれ決められた「値」があり、これを変更することで設定を行います。前述の［BackColor］プロパティでは、色を16進数で指定するようになっています。同様に、［Caption］プロパティでは直接表示したい文字列を設定します。

プロパティは、それぞれのオブジェクトごとに何種類も用意されており、値の設定や読み出し（「参照」といいます）ができるようになっています。プロパティの値の設定は、プロパティウィンドウで行うか、コードで式を作成して行います。また、プロパティはオブジェクトごとに用意されているため、プロパティを操作する場合は、必ずどのオブジェクトのプロパティを変更するのか、対象となるオブジェクトを指定します。

!!! POINT

ラベルコントロールを使って、フォームに文字を表示します。ラベルコントロールは、文字を表示するコントロールです。ラベルコントロールの［Caption］プロパティと［Font］プロパティを使って、表示する文字を設定します。また、［BackColor］［ForeColor］プロパティを操作して、文字色と背景色も設定してみましょう。

コントロールとプロパティの設定

※新しいプロジェクトとフォームを用意します。

1 ツールボックスの［Label］ボタンをクリックする

1 このボタンをクリック

2 フォーム上でドラッグして、ラベルコントロールを配置する

1 ここをクリックして…

2 そのままここまでドラッグ

新しいプロジェクトの作成

新しいプロジェクトとフォームを作成するには、［ファイル］メニューから［新しいプロジェクト］を選択し、［新しいプロジェクト］ダイアログボックスで［標準.EXE］を選択して［OK］をクリックします。

ラベル

プログラムを実行したときに、ユーザーが変更できないテキストのことです。

ドラッグ中は…

マウスポインタをドラッグすると、作成されるコントロールの大きさが白い四角形の枠で表示されます。

3 プロパティウィンドウで［Caption］プロパティの値を「楽しいVisual Basic」に変更する

1 ここをダブルクリック

2 「楽しいVisual Basic」と入力

プロパティ - Label1	
Appearance	1 - 3D
AutoSize	False
BackColor	&H8000000F&
BackStyle	1 - 不透明
BorderStyle	0 - なし
Caption	楽しいVisual Basic
DataField	
DataFormat	

［Caption］の値を選択する

［Caption］をダブルクリックすると、その右側に入力されている値全体が選択されます。または、値をドラッグして選択します。

4 ［BackColor］プロパティをクリックし、右側のセルに表示される下向き三角のボタンをクリックする

1 ここをクリック

2 ここをクリック

プロパティ - Label1	
Appearance	1 - 3D
AutoSize	False
BackColor	&H8000000F&
BackStyle	1 - 不透明
BorderStyle	0 - なし
Caption	楽しいVisual Basic
DataField	
DataFormat	

背景の色を指定する

［BackColor］をクリックすると、その右側のセルの右端に下向き三角のボタンが表示されます。このボタンをクリックすると、カラーパレットが表示されます。

5 カラーパレットが表示される

1 ここをクリック

Appearance 1 - 3D
AutoSize False
BackColor &H00FFFFFF&
パレット システム

タブを切り替える

前回操作したときの状態によっては、［システム］タブが表示されていることがあります。その場合は、［パレット］タブをクリックしてカラーパレットを表示してください。

6 ［パレット］タブで青色を選択する

BackColor　&H00FF0000&
BackSty
BorderS
Caption
DataFie
DataFor
DataMe
パレット　システム
BackCol
オブジェクト…背
景色を設
1 ここをクリック

7 ラベルコントロールの背景が青色に変わる

Form1
楽しいVisual Basic

8 ［ForeColor］プロパティをクリックし、右側のセルに表示される下向き三角のボタンをクリックする

Label1 Label
全体　項目別
DragMode　0 - 手動
Enabled　True
Font　MS Pゴシック
ForeColor　&H00000000&
Height
Index
Left
LinkItem
LinkMod
パレット　システム
ForeCol
オブジェクト…前
景色を設
1 ここをクリック
2 ここをクリック

色が16進数で表示される

選択した色が16進数の値で表されます。青色を表す16進数の値は、「&H00FF0000&」となります。

［ForeColor］を表示する

［ForeColor］が隠れていて見えないときは、プロパティウィンドウの右端にあるスクロールバーを使って、画面をスクロールしてください。

1 - 3D
False
&H8000000F&
1 - 不透明
0 - なし
楽しいVisual Basic

［BackColor］と同様にカラーパレットが表示されます。

9 ［パレット］タブで黄色を選択する

Font	MS Pゴシック
ForeColor	&H0000FFFF&
Height	
Index	
Left	
LinkItem	
LinkMod	

パレット　システム

1 ここをクリック

ForeColor
オブジェクト…
景色を設…

10 ［Font］プロパティをクリックし、右側のセルに表示される［…］ボタンをクリックする

プロパティ－ Label1

Label1 Label

全体　項目別

DragMode	0 － 手動
Enabled	True
Font	MS Pゴシック …
ForeColor	&H0000FFFF&
Height	615
Index	
Left	480

1 ここをクリック

2 ここをクリック

フォント

フォント名(F): MS Pゴシック
MS Pゴシック
MS P明朝
MS Sans Serif
MS Serif
MS ゴシック
MS 明朝
Roman

スタイル(Y): 標準
標準
イタリック
ボールド
ボールド イタリック

サイズ(S): 9
9
10
11
12
14
16
18

OK
キャンセル

文字飾り
取り消し線(K)
下線(U)

サンプル
Aaあぁアァ亜宇

書体の種類(R): 日本語

文字の色が変わる

「楽しいVisual Basic」の文字が黄色に変わります。

黄色を表す16進数の値は、「&H0000FFFF&」となります。

［フォントの指定］ダイアログ

ここでは、ラベルのテキストのフォント、フォントサイズ、文字飾り、スタイルを設定することができます。

11 [サイズ] を「18」に変更し、[OK] をクリックする

1 「18」に変更

2 ここをクリック

フォント
フォント名(F): MS Pゴシック
スタイル(Y): 標準
サイズ(S): 18
OK
キャンセル
MS Pゴシック
MS P明朝
MS Sans Serif
MS Serif
MS ゴシック
MS 明朝
Roman
標準
イタリック
ボールド
ボールド イタリック
11 12 14 16 18 20 22
文字飾り
取り消し線(K)
下線(U)
サンプル
Aaあぁアァ亜宇
書体の種類(R):
日本語

サイズを変更する

サイズボックスに直接「18」と入力することができます。ただし数値の入力は半角文字で行ってください。

12 [Alignment] をクリックし、右側のセルに表示される下向き三角のボタンをクリックする

1 ここをクリック

2 ここをクリック

プロパティ - Label1
Label1 Label
全体 | 項目別

(オブジェクト名)	Label1
Alignment	0 - 左揃え
Appearance	0 - 左揃え
AutoSize	1 - 右揃え
BackColor	2 - 中央揃え
BackStyle	1 - 不透明
BorderStyle	0 - なし
Caption	楽しい Visual Basic
DataField	

配置の一覧

下向き矢印ボタンをクリックすると、配置の一覧が表示されます。

[中央揃え]

[Alignment] は「配置」という意味です。ラベルコントロールに対して、テキストをどのように配置するのかという設定を行います。[左揃え] [右揃え] [中央揃え] の3種類があります。

メニューからプログラムを実行する

[実行] メニューから [開始] を選択してプログラムを実行することもできます。

このプログラムはコードを書いていないので何も動作しません。そのまま終了してください。

13 一覧から [中央揃え] を選択する

プロパティ - Label1

Label1 Label

全体 | 項目別

(オブジェクト名)	Label1
Alignment	2 - 中央揃え
Appearance	0 - 左揃え
AutoSize	1 - 右揃え
BackColor	2 - 中央揃え
BackStyle	1 - 不透明
BorderStyle	0 - なし
Caption	楽しいVisual Basic
DataField	

1 ここをクリック

14 文字がコントロールの中央に配置される

Form1

楽しいVisual Basic

15 ツールバーの [開始] ボタンをクリック (P.8参照) してプログラムを実行する

Form1

楽しいVisual Basic

まとめ

コントロールのプロパティには、外観を決定するプロパティと動作を決定するプロパティの２種類があります。今回操作した外観を決定するプロパティには、コントロールの細かい部分を変更できる様々なプロパティが用意されています。

練習問題

Q1　ラベルコントロールの背景色を緑色（&H0000C00&）に変えてください。

解答は巻末に→

Q2　ラベルコントロールのフォントを「MS明朝」に変えてください。

解答は巻末に→

ワンポイントアドバイス

コントロールのプロパティを変更するときは、必ずフォーム上でそのコントロールを選択状態にします。コントロールを選択することで、プロパティウィンドウに表示されるプロパティが切り替えられます。

コントロールが選択された状態になると、コントロールの周囲に黒い四角形が表示され、プロパティウィンドウには「Label1」と表示されます。

Lesson 4

第１日 ２時限目

はじめてのプログラムを作る

Lesson4.vbp
VB6CD
Lesson04

◎Learning Edition

プロパティへの値の代入

マウスやメニューではなく、コードでコントロールのプロパティを変更する方法を覚えます。

今回作成するプログラム

Form1

楽しいVisual Basic

ピンク　緑

1 ［ピンク］［緑］いずれかのボタンをクリック

Form1

楽しいVisual Basic

ピンク　緑

2 ラベルの背景がボタンの色に変更される

POINT

　コードの命令から、コントロールのプロパティを切り替えるテクニックを使って、コマンドボタンでラベルコントロールのプロパティを操作します。ラベルコントロールの［BackColor］プロパティの設定を変えて、コントロールの背景色を切り替える操作を、コードを使って行います。

コントロールとプロパティの設定

※引き続き、「Lesson3.vbp」を使います。

1 ツールボックスの［CommandButton］をクリックする

1 このボタンをクリック

2 フォーム上でドラッグして、コマンドボタンを配置する

1 ここをクリックして…

2 そのままここまでドラッグ

3 プロパティウィンドウで［Caption］プロパティの値を「ピンク」に変更する

1 ここをダブルクリック

2 「ピンク」と入力

Lesson3.vbpファイルの保存場所

このファイルは、付録CD-ROMの「Lesson03」フォルダに収録されています。

ボタンを作成するもう１つの方法

ツールボックスの［CommandButton］ボタンをダブルクリックするとフォームの中央付近に、自動的にボタンが作成されます。後はハンドルを使ってサイズを調節するだけです。

この方法はボタン以外のコントロールにも応用可能です。

ボタンを選択

この操作を行う前に、作成した［Command1］ボタンが選択されていることを確認してください。

4 [(オブジェクト名)] プロパティの値を「Pink」に変更する

プロパティ - Command1	
Command1 CommandButton	
全体	項目別
(オブジェクト名)	Pink
Appearance	1 - 3D
BackColor	&H8000000A&
Cancel	False
Caption	ピンク
CausesValidation	True
Default	False
DisabledPicture	(なし)

1 ここをダブルクリック

2 「Pink」と入力

5 ツールボックスの [CommandButton] をクリックし、もう1つコマンドボタンを配置する

Form1

楽しいVisual Basic

ピンク

Command2

1 ここをクリックして…

2 そのままここまでドラッグ

6 プロパティウィンドウで [Caption] プロパティの値を「緑」に変更する

プロパティ - Command2	
Command2 CommandButton	
全体	項目別
(オブジェクト名)	Command2
Appearance	1 - 3D
BackColor	&H8000000A&
Cancel	False
Caption	緑
CausesValidation	True
Default	False
DisabledPicture	(なし)

1 ここをダブルクリック

2 「緑」と入力

オブジェクト名の変更

オブジェクト名を「Pink」にするのは、コードを記述する際にプロシージャを参照しやすくするためです。

(オブジェクト名) Pink

Pink Click

ボタンを選択

この操作を行う前に、作成した[Command2]ボタンが選択されていることを確認してください。

7 [(オブジェクト名)] プロパティの値を「Green」に変更する

プロパティ－Command2

Command2 CommandButton

全体 | 項目別

(オブジェクト名)	Green
Appearance	1 － 3D
BackColor	&H8000000A&
Cancel	False
Caption	緑
CausesValidation	True
Default	False

1 ここをダブルクリック

2 「Green」と入力

コードの記述

Form1:Pink-Click

Project1 － Form1 (コード)

Pink | Click

```
Private Sub Pink_Click()
    Label1.BackColor = &HFF80FF
End Sub
```

Form1:Green-Click

Project1 － Form1 (コード)

Green | Click

```
Private Sub Green_Click()
    Label1.BackColor = &HC000&
End Sub
```

ワンポイントアドバイス

最初に［ピンク］ボタンをダブルクリックしてコードウィンドウを表示し、コードを入力した後、続けて［緑］ボタンをダブルクリックしてコードウィンドウを表示し、コードを入力します。

解説

1 ［ピンク］ボタンをダブルクリックして、このボタンのClickイベントプロシージャを表示します。プロシージャ「Sub Pink_Click()」が、このボタンが押されたときに実行されるプロシージャです。

2 このボタンは、ラベルコントロールの背景色をピンク色にします。背景色を変えるには、［BackColor］プロパティに色を数値（16進数）で設定します。

3 プロシージャにコントロール名「Label1」を入力します。
コントロール名は、そのコントロールのプロパティウィンドウにある「オブジェクト名」（ここでは「Label1」）が、そのままコントロール名になります。

4 「Label1」に続いて「.」を入力します。自動的に、「Label1」で使えるプロパティがリスト表示されます。

```
Project1 - Form1 (コード)
Green                    Click
Private Sub Green_Click()
    Label1.
End Sub
        Alignment
        Appearance
        AutoSize
        BackColor
        BackStyle
        BorderStyle
        Caption
```

使用できるプロパティのリスト

5 プロパティ［BackColor］をマウスでダブルクリックするか、Tabキーを押せば、そのままコードに入力されます。続いて、値を設定するために「=」を入力し、そのあとにピンク色の16進コード「&HFF80FF」を入力します。

6 同様に、［緑］ボタンのプロシージャ「Sub Green_Click()」も同じように記述していきます。ただしプロパティ［BackColor］の値は、「&HC000&」に設定します。

7 コードを記述したら、プログラムを実行します。［ピンク］ボタンを押すとラベルコントロールの背景色がピンク色に、［緑］ボタンを押すと、緑色に変わります。

8 プロパティに値を設定するときは、「=」を使います。これは、「Let」ステートメントといい、=の右辺を左辺に代入する働きをします。本来は、

```
Let Label1.BackColor = &HC000&
```

と記述するのですが、通常は「Let」を省略して「=」だけで記述します。

まとめ

コードでコントロールのプロパティを操作するときは、必ずコントロール名（オブジェクト名）とプロパティを「.」（ピリオド）で指定します。常に、どのコントロールのどのプロパティを操作するのかを明確にしておかなければなりません。

Label1.BackColor = &HC000&

- Label1：コントロール名（オブジェクト名）
- .：ピリオド
- BackColor：プロパティ
- =：イコール（Let ステートメント）
- &HC000&：値

練習問題

Q1　コマンドボタンを1つ追加し、ボタンを押すとラベルコントロールの背景色が「青色」になるようにしてください。

解答は巻末に→

Q2　［ピンク］ボタンを押したときに、文字の色をピンクに変えるように変更してください。

【ヒント】
文字の色は［ForeColor］プロパティが担当します。

解答は巻末に→

ワンポイントアドバイス

プロパティ「BackColor」は、ほとんどのコントロールとフォームが持っているプロパティです（コントロールによっては背景色の設定ができないものもあります）。

プロパティの値は、色の番号を16進数で指定します。プロパティウィンドウのカラーパレットを使うと、色番号を知ることができます。

ForeColor　&H0000FFFF&
HasDC
Height
HelpConte
Icon
KeyPreviev
パレット　システム

この数値が色の16進数番号

Lesson 5

第1日
3時限目

はじめてのプログラムを作る

Lesson5.vbp
VB6CD
Lesson05

◎Learning Edition

イメージファイルを使う

イメージコントロールに画像ファイルを設定する方法を覚えます。フォームのプロパティも操作します。

今回作成するプログラム

Form1

指定された画像ファイルが表示される

終了

ボタンをクリックすると…

プログラムが終了する

POINT

イメージを表示するだけのプログラムです。付録CD-ROMの「Lesson05」フォルダにある「Mark.bmp」というビットマップファイルを使います。

コントロールとプロパティの設定

※新しいプロジェクトとフォームを用意します。

1 ツールボックスの［Image］ボタンをクリックする

1 このボタンをクリック

2 フォーム上でドラッグして、コントロールを配置する

1 ここをクリックして…

2 そのままここまでドラッグ

3 プロパティウィンドウで［Picture］プロパティをクリックし、右側のセルのボタンをクリックする

プロパティ - Image1	
MouseIcon	(なし)
MousePointer	0 - 既定値
OLEDragMode	0 - 手動
OLEDropMode	0 - なし
Picture	(なし)
Stretch	False

1 ここをクリック

2 ここをクリック

新しいプロジェクトの作成

新しいプロジェクトとフォームを作成するには、［ファイル］メニューから［新しいプロジェクト］を選択し、［新しいプロジェクト］ダイアログボックスで［標準.EXE］を選択して［OK］をクリックします。

イメージコントロールとは

ビットマップや画像をフォーム上に表示するコントロールです。

イメージを選択

この操作を行う前に、作成したイメージコントロールが選択されていることを確認してください。

「Mark.bmp」の保存場所

このファイルは、付録CD-ROMの「Lesson05」フォルダに収録されています。

フォームを選択する

この操作を行う前に、フォームを選択してください。フォームを選択するには、フォーム上のイメージコントロール以外の部分をクリックします。

フォームのまわりにハンドルが表示されていると選択されたことになります。

4 [ピクチャの読み込み] ダイアログが表示されるので、[Mark.bmp] を選択し、[開く] をクリックする

1 [Mark.bmp]を選択

2 ここをクリック

5 イメージコントロールに選択した画像が表示される

6 フォームを選択してから [BackColor] プロパティをクリックし、右側のセルのボタンをクリックする

1 ここをクリック

2 ここをクリック

7 カラーパレットが表示されるので、［パレット］タブで「白色」を選択する

BackColor　&H00FFFFFF&
BorderSt
Caption
ClipCont
ControlB
DrawMod
DrawSty
DrawWid
パレット　システム
BackCol
オブジェクト
景色を設

1 ここをクリック

8 イメージコントロールの背景が白くなる

Form1

背景が白くなった

9 ツールボックスの［CommandButton］をクリックし、コマンドボタン１つを配置する

Form1

Command1

1 ここをクリックして…

2 そのままここまでドラッグ

ボタンを選択

この操作を行う前に、作成した［Command1］ボタンが選択されていることを確認してください。

10 プロパティウィンドウで［Caption］プロパティの値を「終了」に変更する

プロパティ - Command1

Command1 CommandButton

全体 | 項目別

(オブジェクト名)	Command1
Appearance	1 - 3D
BackColor	&H8000000F&
Cancel	False
Caption	終了
CausesValidation	True
Default	False
DisabledPicture	(なし)
DownPicture	(なし)
DragIcon	(なし)
DragMode	0 - 手動

1 ここをダブルクリック

2 「終了」と入力

コードの記述

Form1:Command1-Click

Project1 - Form1 (コード)

Command1 ▼　Click ▼

```
Private Sub Command1_Click()
    End
End Sub
```

解説

1 　コマンドボタンをダブルクリックして、Clickイベントプロシージャを表示します。そして、プログラム終了の「End」ステートメントを記述します。

まとめ

イメージコントロールに画像ファイルを設定する場合は、［Picture］プロパティにその画像ファイルを指定します。イメージコントロールが表示できるイメージファイルの種類は以下のとおりです。

拡張子	種類
bmp	Windows ビットマップファイル
gif	compuserve GIF ファイル
jpg	JPEG ファイル
wmf/emf	Windows（拡張）メタファイル
ico	Windows アイコンファイル
cur	Windows カーソルファイル

練習問題

Q1 イメージファイルを「Free.bmp」に入れ替えてください。ファイルは付録 CD-ROM の「Lesson05」フォルダに収録してあります。

解答は巻末に→

ワンポイントアドバイス

イメージコントロールの［Stretch］プロパティは、表示する画像をイメージコントロールのサイズに設定します。

プロパティの値が［False］で、設定した画像ファイルはそのままのサイズで表示されます。［True］にすると、画像ファイルはイメージコントロールの大きさに強制的に合わせられます。

Trueのとき

Falseのとき

Lesson 6

第１日
３時限目

はじめてのプログラムを作る

Lesson6.vbp
VB6CD
Lesson06

◎Learning Edition

イメージを切り替える

イメージコントロールの画像ファイルをコードで指定する方法を覚えます。また、イベントプロシージャを使ったテクニックを覚えます。

● 今回作成するプログラム

Form1

終了

1 イメージをクリックすると…

Form1

フリー素材集

終了

2 イメージが「Free.bmp」に切り替わる

POINT

コマンドボタンを使って、イメージコントロールに表示するイメージを切り替えます。ただし、コマンドボタンを使わずに、イメージをクリックすると切り替わるようにします。

コントロールとプロパティの設定

※ここでは、前のレッスンで使用した「Lesson5.vbp」ファイルを引き続き使用します。「Free.bmp」をＣドライブのルートにコピーしておいてください。

1 イメージコントロールをダブルクリックしてコードウィンドウを開く

1 ここをダブルクリック

コードの記述

Form1:Image1-Click

Project1 - Form1 (ｺｰﾄﾞ)

Image1 / Click

```
Private Sub Image1_Click()
    Image1.Picture = LoadPicture("c:¥Free.bmp")
End Sub
```

解説

1

イメージコントロールがクリックされると、コマンドボタンと同様にClickイベントが発生します。このときに実行したい処理は、このプロシージャに記述します。

ここでは、イメージコントロール「Image1」に発生したClickイベントなので、「Image1_Click()」プロシージャに、イメージファイルを入れ替える処理を記述します。

2

イメージコントロールに新しいイメージファイルを表示させるには、[Picture] プロパティにそのイメージファイル名を設定します。

ただし、直接イメージファイル名を記述するのではなく、LoadPicture関数を使って、イメージファイルを設定します。

```
Image1.Picture = LoadPicture("c:¥Free.bmp")
```

関数（LoadPicture）　引数（ロードするファイルの名前）（"c:¥Free.bmp"）

【LoadPicture関数の書式】

```
LoadPicture(filename)
```

filename	ロードする画像ファイルのファイル名をパス付きで記述します。

3

LoadPicture関数は、() 内に指定したイメージファイルを、イメージコントロールにロードする役割を果たします。この関数の戻り値を、イメージコントロールの [Picture] プロパティに代入します。

ちなみに、LoadPicture関数を使わずにファイル名を指定するとエラーになります。なぜ、ファイル名（パス名）を直接指定してはいけないのでしょうか？ それは、ファイル名を示す文字列というのは、単なるファイルの「場所情報」だからです。イメージコントロール内に場所情報をロードすることはできませんから、当然エラーになります。

イメージコントロールに画像を表示するには、場所情報ではなく、イメージファイルの「本体」をロードする必要があります。この作業を受け持って実行してくれるのがLoadPicture関数です。LoadPicture関数にファイルの場所を指定してやれば、変換された値がイメージコントロールの [Picture] プロパティに代入され、画像が表示されます。

指定するファイル名は、" "（ダブルクォーテーション）で囲んで指定します。このとき、ファイル名の指定はパス名を付けて指定してください。拡張子は必ず付けます。絶対パスで参照するときは「c:¥Free.bmp」などのようにドライブ名やフォルダ名をきちんと記述してください。

4

プログラムを実行して、イメージをクリックしてください。表示している画像が、指定したイメージファイルに入れ替わります。

まとめ

プロパティには、直接値を設定するものもあれば、このように関数やメソッド（P.79参照）を使って設定するものもあります。

また、これから出てくる「変数」（P.70参照）を使ってプロパティを設定する手法は、よく使われるテクニックです。

練習問題

Q1 「Free.bmp」を任意のフォルダにコピーして、LoadPicture関数から指定してください。

解答は巻末に→

ワンポイントアドバイス

ドライブ名とフォルダ名をつないだ記述を、「パス」と呼びます。MS-DOSを使っていた人にはおなじみの表現ですね。Visual Basicでもまったく同じ記述方法を採用しています。

パスを記述する際は、ドライブ名とフォルダ名は「:¥」で、フォルダ名とファイル名は「¥」記号でつなぎます。例えば、ドライブCの「data」フォルダにある「girl.bmp」ファイルをフルパスで記述すると、次のようになります

```
c:¥data¥girl.bmp
```

COLUMN

ステートメントについて

「ステートメント」とは、プログラム作成を行う上で必要な命令のことで、プログラムの流れを制御したり、コードを書きやすくするために用意されています。日本語では「～文」とも呼ばれます。例えば、後述のレッスンで出てくる、For...Nextステートメントは、指定した回数だけ、For...Next間の処理を繰り返すステートメントです。Ifステートメントは、ある条件を判断して、プログラムの流れを振り分けます。

Visual Basicのプログラム作成は、これらのステートメントを組み合わせながら、オブジェクトのプロパティやメソッドを操作して行います。ステートメントの種類はたくさんあるのですが、本書に出てくるような基本的なステートメントだけでも、かなり面白い処理をすることができます。必要に応じて覚えていくようにすればよいでしょう。

Lesson 7

第１日
４時限目

はじめてのプログラムを作る

Lesson7.vbp
VB6CD
Lesson07

◎Learning Edition

コントロールを組み合わせる

これまで紹介したコントロールを使って、フォームをデザインします。
また、フレームコントロールを使ってフォームを飾ります。

● 今回作成するプログラム

Form1

よくわかるVisual Basic6.0

By Seto(●^_^●) 1998.11.

OK

このボタンをクリックすると…

情報を表示するダイアログボックスが閉じる

POINT

これまで使ったコントロールと新しい「フレーム」コントロールを組み合わせて、画面に情報を表示するプログラムを作成します。
この形式のプログラムは、「バージョン情報」のダイアログボックスなどによく使われています。

コントロールとプロパティの設定

※新しいプロジェクトとフォームを用意します。

1 ツールボックスの［Frame］ボタンをクリックする

1 このボタンをクリック

2 フォーム上でドラッグして、フレームコントロールを配置する

1 ここをクリックして…

2 そのままここまでドラッグ

新しいプロジェクトの作成

新しいプロジェクトとフォームを作成するには、［ファイル］メニューから［新しいプロジェクト］を選択し、［新しいプロジェクト］ダイアログボックスで［標準.EXE］を選択して［OK］をクリックします。

フレームコントロール

フレームコントロールは複数のコントロールをグループにして、他のコントロールと区別するために使います。フレームを移動すると、フレーム上のコントロールはすべて同時に移動します。フレームコントロールは、このように複数のコントロールをまとめる働きをします。

フレームを選択する

この操作を行う前に、作成したフレームコントロールが選択されていることを確認してください。

キャプションは無くてもよい

フレームなどのコントロールで特に［Caption］が必要ないときは削除してしまってもかまいません。フォーム上にキャプションが表示されなくなります。

3 プロパティウィンドウでフレームコントロールの［Caption］プロパティの値を削除する

1 ここをダブルクリック

2 Del キーを押して「Frame1」を削除

4 ツールボックスの［Image］ボタンをクリックする

1 このボタンをクリック

5 フレーム上にイメージコントロールを配置する

1 ここをクリックして…

2 そのままここまでドラッグ

6 プロパティウィンドウで［Picture］プロパティをクリックし、右側のセルの［...］ボタンをクリックする

1 ここをクリック

2 ここをクリック

7 ［ピクチャの読み込み］ダイアログで［angel.gif］を選択し、［開く］をクリックする

1 ［angel.gif］を選択

2 ここをクリック

コントロールを重ねる

既に作成されているコントロールの上に、別のコントロールを重ねて作成することができます。

フレームコントロールの特徴

フレームコントロールに他のコントロールを載せる場合は必ず、フレームを先に配置してから、その上に他のコントロールを配置します。そうしないと、フレームにコントロールが隠れてしまいます。

「angel.gif」ファイルの保存場所

「angel.gif」ファイルは、付録CD-ROMの「Lesson07」フォルダに収録されています。

イメージコントロールのサイズ

イメージコントロールのサイズは、選択したイメージのサイズに合わせて調整されます。(P.41参照)

8 選択したイメージがイメージコントロールに表示される

Form1

9 ツールボックスで［Label］ボタン A をクリックし、フレーム上にラベルコントロールを配置する

Label1

1 ここをクリックして…

2 そのままここまでドラッグ

10 プロパティウィンドウで［Caption］プロパティの値を変更する

プロパティ - Label1

Label1 Label

全体 | 項目別

(オブジェクト名)	Label1
Alignment	0 - 左揃え
Appearance	1 - 3D
AutoSize	False
BackColor	&H8000000F&
BackStyle	1 - 不透明
BorderStyle	0 - なし
Caption	よくわかるVisual Basic6.0
DataField	
DataFormat	

1 ここをダブルクリック

2 「よくわかるVisual Basic6.0」と入力

11 ラベルコントロールのサイズを調整する

Form1

よくわかるVisual Basic6.0

1 黒い四角形をドラッグ

12 [Label] ボタンを使って、フレーム上にラベルコントロールをもう1つ配置する

Form1

よくわかるVisual Basic6.0

Label2

1 ここをクリックして…

2 そのままここまでドラッグ

13 手順10と11を参考にして、[Caption] プロパティの値を変更し、ラベルのサイズを調整する

Form1

よくわかるVisual Basic6.0

By Seto(●^_^●) 1998.11.

1 このように変更する

サイズ変更ハンドル

コントロールを選択したときに、その周囲に表示される黒い四角形を「サイズ変更ハンドル」と呼びます。

このハンドルをドラッグして、コントロールのサイズを変更します。

14 ツールボックスの［CommandButton］ を クリックし、コマンドボタンを１つ配置する

Form1

よくわかるVisual Basic6.0

By Seto(●^_^●) 1998.11.

Command1

1 ここをクリックして…

2 そのままここまでドラッグ

15 プロパティウィンドウで［Caption］プロパティの値を「OK」に変更する

プロパティ－Command1

Command1 CommandButton

全体 | 項目別

(オブジェクト名)	Command1
Appearance	1 － 3D
BackColor	&H8000000F&
Cancel	False
Caption	OK
CausesValidation	True
Default	False
DisabledPicture	(なし)

1 ここをダブルクリック

2 「OK」と入力

コードの記述

Form1:Command1-Click

Project1 － Form1 (コード)

Command1 | Click

```
Private Sub Command1_Click()
    End
End Sub
```

解説

1　コマンドボタンのClickイベントプロシージャに、もうすっかりおなじみのプログラム終了ステートメント「End」を記述します。

まとめ

イメージコントロール、ラベルコントロール、フレームコントロール、コマンドボタンを組み合わせてフォームに配置すると、アプリケーションなどでおなじみの、「バージョン情報（...について）」ダイアログボックスができあがります。

このレッスンで学んだ、コントロールを組み合わせてフォームを飾るテクニックは、いろいろと使い道があるので、アイデアを取り入れて自分のお気に入りのフォームにしてください。

練習問題

Q1　フレーム、フォーム、ラベルコントロールの背景色を、ピンクに変えてください。

解答は巻末に→

ワンポイントアドバイス

ラベルコントロールの［BackStyle］プロパティを「0－透明」に設定すると、文字だけが表示されるので、イメージとラベルコントロールを重ね合わせることができます。

Form1
Welcome! Hot Cocoa
OK!

もちろん、ラベルコントロール同士も重ね合わせることができます。

左図のフォームでは、「Welcome! Hot Cocoa」という色の違う文字（ラベルコントロール）を２つ重ねて、影付き文字の効果を出しています。ラベルコントロールの背景を透明にすれば、こういう効果も作れます。

Let's VBSchool!!

第2日 (Lesson8 ~ Lesson12)
イベントを使う

Windowsのプログラムの大きな特徴は、「イベント駆動型」のウィンドウシステムであることです。

ユーザーが操作するメニューやボタン、マウスの動作などを捕らえて、そのイベントに対応した処理をプログラム側で組み上げていくのです。

本来は、複雑な処理が必要とされるこのシステムなのですが、Visual Basicでは簡単にイベント処理ができるように作成されています。

第２日は、このイベントに対する理解と、イベントに対応した処理を組むにはどうすれば良いのかをテーマに解説します。

今日のレッスンで解説する項目

- ❏ ClickイベントとDblClickイベントという、２つのイベントの使い分け
- ❏ それぞれのイベントプロシージャの記述方法
- ❏ 「MouseDown」イベントの使い方
- ❏ イベントプロシージャで扱う変数の使い方と「変数」についての理解
- ❏ 複数のイベントプロシージャ間で１つの変数を共有する方法
- ❏ MouseDownとMouseUpイベントの組み合わせでプロシージャを組み立てる
- ❏ MouseMoveイベントの使い方

Lesson
8

第２日
１時限目

イベントを
使う

Lesson8.vbp
VB6CD
Lesson08

◎Learning Edition

クリックとダブルクリック

１つのコントロールでの、Click イベントと DblClick イベントの使い分けと、それぞれのイベントプロシージャの記述方法を覚えます

● 今回作成するプログラム

Form1

イメージをクリック
またはダブルクリック

Form1

クリックすると枠線
が表示される

lesson8
ダブルクリックされました
OK

ダブルクリックすると
メッセージが表示される

POINT

Windows プログラミングの特徴の１つ、イベント処理について理解するために、なじみの深いイベントであるマウスの「クリック」と「ダブルクリック」を使ったプログラムを作成します。

このプログラムは、イメージコントロールをクリックした場合とダブルクリックした場合で、別々の処理を行います。

コントロールとプロパティの設定

※新しいプロジェクトとフォームを用意します。

1 ツールボックスの［Image］ボタンをクリックする

Image

1 このボタンをクリック

2 フォーム上でドラッグして、コントロールを配置する

Form1

1 ここをクリックして…

2 そのままここまでドラッグ

3 プロパティウィンドウで［Picture］プロパティに「angel.gif」ファイルを設定する

プロパティ－Image1

Image1 Image

全体 | 項目別

MouseIcon	(なし)
MousePointer	0 - 既定値
OLEDragMode	0 - 手動
OLEDropMode	0 - なし
Picture	(なし)
Stretch	False
Tag	

ピクチャの読み込み

ファイルの場所(I): lesson08

angel.gif

ファイル名(N): angel.gif

ファイルの種類(T): すべてのピクチャ ファイル

開く(O)　キャンセル　ヘルプ(H)

1 ここをクリック

2 ここをクリック

3 ここをクリック

4 ここをクリック

イメージを選択

この操作を行う前に、作成したイメージコントロールが選択されていることを確認してください。

「angel.bmp」の保存場所

このファイルは、付録CD-ROMの「Leeson08」フォルダに収録されています。

4 イメージコントロールに選択した画像が表示される

コードの記述

Form1:Image1-Click

Project1 - Form1 (コード)

Image1 / Click

```
Private Sub Image1_Click()
    Image1.BorderStyle = 1
End Sub
```

Form1:Image1-DblClick

Project1 - Form1 (コード)

Image1 / DblClick

```
Private Sub Image1_DblClick()
    MsgBox "ダブルクリックされました"
    Image1.BorderStyle = 0
End Sub
```

ワンポイントアドバイス

コードウィンドウで異なるプロシージャを切り替えるには、プロシージャメニューを使って、コードを記述するプロシージャを選択します。

解説

1 まず、フォームに配置したイメージコントロールをダブルクリックして、コードウィンドウを表示します。コードウィンドウには、イメージコントロールのClickイベントプロシージャが表示されます。

2 ユーザーによってウィンドウの部品が操作されたとき、その操作を「イベント」と呼びます。イベントは何種類もあり、それぞれ名前が付いています。例えば、マウスがクリックされれば「Click」イベントが、キーボードのキーが押されれば「KeyDown」イベントが発生します。

Visual BasicによるWindowsのプログラムは、このユーザーが起こしたイベントに対応して、プログラムの処理を作成していきます。

3 イベントが発生すると、そのイベントと同じ名前のプロシージャが呼び出され、プロシージャに記述したコードが実行されます。これを、「イベントプロシージャ」と呼びます。イベントプロシージャは、そのオブジェクト名と必ずペアになってプロシージャが形成されています（P.11参照）。

例えば、コマンドボタンが押されれば、コマンドボタンのオブジェクト名「CommandButton1」とイベント名「Click」を「_」（アンダースコア）でつないだ名前「CommandButton1_Click()」というイベントプロシージャが呼び出されます。

```
CommandButton1_Click()
```

オブジェクト名　イベント名

4 フォームに配置したイメージコントロールが、ユーザーによってクリックされれば、「Image1_Click()」というプロシージャが呼び出されます。この時に、なにか処理を実行したければ、このプロシージャに処理を記述します。これで、ユーザーがイメージコントロールをクリックするたびに、Clickイベントプロシージャが実行されます。

5 本プログラムでは、イメージコントロールがクリックされると、イメージ全体に枠線を付けます。

Image1_Click()プロシージャに、以下のコードを記述します。

```
Image1.BorderStyle = 1
```

この記述は、イメージコントロールの［BorderStyle］プロパティを、「1 - 実線」に設定します。このプロパティが設定されると、イメージコントロールは枠線で囲まれて表示されます。

BorderStyleが１の
表示状態

6　次に、イメージコントロールがユーザーによってダブルクリックされた場合の処理を作成します。

ダブルクリックされた場合は、Clickイベントプロシージャではなく、ダブルクリックされたときのイベントプロシージャ「DblClick」に、その処理を記述します。

7　コードウィンドウでDblClickイベントプロシージャを作成するには、コードウィンドウのイベント用リストボックスをクリックして、イベントリストを表示します。そして、リストの中から、「DblClick」イベントを選択します。

Project1 - Form1 (ｺｰﾄﾞ)

Image1 | Click

```
Private Sub Image1_Click()
    Image1.BorderStyle = 1
End Sub
```

Click
DblClick
DragDrop
DragOver
MouseDown
MouseMove
MouseUp
OLECompleteDrag
OLEDragDrop
OLEDragOver
OLEGiveFeedback
OLESetData

イベント用リスト
ボックス

8　イベントが選択されると、自動的にコードウィンドウにそのイベント・プロシージャが挿入されます。

このとき、プロシージャ名は「Image1_DblClick()」になります。

Project1 - Form1 (ｺｰﾄﾞ)

Image1 | DblClick

```
Private Sub Image1_Click()
    Image1.BorderStyle = 1
End Sub

Private Sub Image1_DblClick()

End Sub
```

9　このプログラムでは、ダブルクリックイベントが発生すると、メッセージボックスを表示し、イメージコントロールの［BorderStyle］プロパティを「0 - なし」に設定して枠線を消します。この処理を、DblClickイベントプロシージャに記述します。

```
Private Sub Image1_DblClick()
    MsgBox "ダブルクリックされました"
    Image1.BorderStyle = 0
End Sub
```

10　プログラムを実行します。イメージをクリックすると、イメージの周りに3D表示の枠線が設定されます。イメージをダブルクリックすると、メッセージボックスが表示され、イメージの枠線が消えます。

まとめ

今回は、1つのコントロールに2つのイベント処理を作成しました。イベントプロシージャは、フォームやコントロール（オブジェクト）に対応して、それぞれのオブジェクトによって異なるプロシージャが設定されています。

例えば、イメージコントロールにはDblClickイベントがありますが、コマンドボタンコントロールにはありません。なぜならコマンドボタンには「ダブルクリック」という動作がないからです。

それぞれのオブジェクトが、どのようなイベントに対応しているのかは、コードウィンドウのイベントリストボックスを見ればわかります。

練習問題

Q1　フォームをダブルクリックすると、フォームの背景色が黄色（&HFFFF&）になるようにしてください。

解答は巻末に→

ワンポイントアドバイス

「Lesson8.vbp」を実行して、最初にイメージをダブルクリックすると、クリックイベントの処理もいっしょに動作してしまいます（BorderStyleが1になった後、メッセージが表示される）。

これは、ダブルクリックがクリック動作も内包しているためで、実際のプログラム作成では、クリックとダブルクリックの2つの処理を1つのコントロール内にいっしょに記述することは、避けたほうがいいでしょう。

また、プロシージャからプロパティを操作する場合、プロパティによっては値の取得しかできず、値の設定ができないものもあるので、注意してください。

Lesson 9

第2日 2時限目

イベントを使う

Lesson9.vbp
VB6CD
Lesson09

◎Learning Edition

マウスイベントを使う

マウスイベントの1つ「MouseDown」イベントの使い方とコードからコントロールのプロパティを操作する方法を覚えます。

● 今回作成するプログラム

マウスイベント－MouseDown

Welcome!

1 任意の位置をクリックすると…

マウスイベント－MouseDown

Welcome!

2 その位置にラベルが移動する

POINT

ここからのレッスンでは、数あるイベント処理の中でも応用範囲の広い、マウスのイベントを使ったプロシージャの作成を勉強していきます。このプログラムは、フォーム上でマウスでクリックすると、「MouseDown」イベントを検出して、クリックした位置にラベルコントロールが移動するプログラムです。

コントロールとプロパティの設定

※新しいプロジェクトとフォームを用意します。

1 Form1の［Caption］プロパティの値を「マウスイベントー MouseDown」に変更する

プロパティ - Form1

Form1 Form

全体 | 項目別

(オブジェクト名)	Form1
Appearance	1 - 3D
AutoRedraw	False
BackColor	&H8000000F&
BorderStyle	2 - 可変
Caption	マウスイベントー MouseDown
ClipControls	True
ControlBox	True

1 ここをダブルクリック

2 この値を変更

2 フォームのCaptionが変わる

マウスイベントー MouseDown

「マウスイベントー MouseDown」に変わる

3 ツールボックスの［Label］ A をクリックし、フォーム上でドラッグしてラベルコントロールを配置する

マウスイベントー MouseDown

Label1

1 ここをクリックして…

2 そのままここまでドラッグ

フォームを選択

新しいプロジェクトを作成した直後は、フォームが選択されています（フォームのハンドルが表示状態）。そのままプロパティを変更してください。

ツールボックスのボタン名

ツールボックスのボタンの上にマウスポインタを移動すると、そのボタンの名前が表示されます。

ラベルを選択

この操作を行う前に、作成したラベルコントロールが選択されていることを確認してください。

4 コードウィンドウで［Caption］プロパティの値を「Welcome!」に変更する

プロパティ - Label1	
Label1 Label	
全体	項目別
AutoSize	False
BackColor	&H8000000F&
BackStyle	1 - 不透明
BorderStyle	0 - なし
Caption	Welcome!
DataField	
DataFormat	
DataMember	

1 ここをダブルクリック

2 「Welcome!」と入力

5 ［Font］プロパティの［…］ボタンをクリックし、フォントサイズを「14」に変更する

プロパティ - Label1	
Label1 Label	
全体	項目別
DataMember	
DataSource	
DragIcon	(なし)
DragMode	0 - 手動
Enabled	True
Font	MS Pゴシック
ForeColor	&H000000FF&
Height	360

1 ここをクリック

2 ここをクリック

フォント

フォント名(F): MS Pゴシック
Marlett
Modern
MS Pゴシック
MS P明朝
MS Sans Serif
MS Serif
MS ゴシック

スタイル(Y): 標準
標準
イタリック
ボールド
ボールド イタリック

サイズ(S): 14
10
11
12
14
16
18
20

OK
キャンセル

文字飾り
取り消し線(K)
下線(U)

サンプル
Aaあぁアァ亜宇

書体の種類(R): 日本語

3 「14」に変更

4 ここをクリック

6 ラベルコントロールを文字のサイズに合わせて縮小する

マウスイベント－MouseDown

Welcome!

1455 x 360

1 サイズ変更ハンドルをドラッグ

サイズ変更ハンドル

コントロールのサイズを変更するには、コントロールを選択したときに、その周囲に表示されるサイズ変更ハンドル（黒い四角形）をドラッグします。

7 [ForeColor] プロパティ（文字色）の [▼] ボタンをクリックし、赤（&H000000FF&）に変更する

プロパティ－Label1

Label1 Label

全体　項目別

DataSource	
DragIcon	(なし)
DragMode	0 － 手動
Enabled	True
Font	MS Pゴシック
ForeColor	&H000000FF&
Height	360
Index	
Left	600

1 ここをクリック

2 ここをクリック

全体　項目別

Font	MS Pゴシック
ForeColor	&H000000FF&
Height	
Index	
Left	
LinkItem	
LinkMode	
LinkTimeou	
LinkTopic	
MouseIcon	
MousePoir	

パレット　システム

ForeColor

オブジェクト内の文字やグラフィックの表示で使用する前景

3 ここをクリック

4 ここをクリック

コードの記述

Form1:Form-MouseDown

Project1 - Form1 (ｺｰﾄﾞ)

Form ▼ | MouseDown ▼

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, _
                        X As Single, Y As Single)
    If Button = 1 Then
        Label1.Left = X
        Label1.Top = Y
    End If
End Sub
```

解説

1 このプログラムは、フォームをマウスでクリックすると、クリックした位置にラベルコントロールが移動するプログラムです。

使用するイベントプロシージャは、フォームのMouseDownイベントプロシージャです。

2 配置したフォームをマウスでダブルクリックすると、コードウィンドウが開きます。ここには、フォームのLoadイベント・プロシージャが表示されていますので、イベントリストボックスから「MouseDown」イベントを選びます。

3 コードウィンドウに、このMouseDownイベントプロシージャ「Form_MouseDown」が挿入されます。

MouseDownイベントは、ユーザーがマウスのボタンを押したときに発生するイベントです。「Click」イベントとの違いは、クリックが「ボタンを押して離す」動作を連続して行ったときに発生するのに対し、MouseDownイベントはマウスのボタンを押しているときに発生するイベントであることです。

さらに、このイベントプロシージャは、実行されると押されたボタンの種類や、ボタンが押された場所のXY座標位置などの情報を、変数（下のコラム参照）に格納します。

【MouseDownイベントプロシージャの書式】

```
Private Sub Form_MouseDown(Button As Integer, _
      Shift As Integer, X As Single, Y As Single)
```

Button As Integer

マウスのボタンが押されたときの、ボタンの種類が変数Buttonに格納されます。

1－左ボタン
2－右ボタン
4－中央ボタン（3ボタンの場合）

Shift As Integer

マウスのボタンが押されたときに、いっしょに押されていたキーの種類が変数Shiftに格納されます。複数のキーが同時に押された場合は、それぞれの値の和が格納されます。たとえばCtrlキーとShiftキーを同時に押したときは「3」が格納されます。

1－Shiftキー
2－Ctrlキー
4－Altキー

X As Single, Y As Single

マウスのボタンが押された時の、カーソル位置のX座標が変数Xに、Y座標が変数Yにそれぞれを格納されます。

COLUMN ・・・・・

変数と引数

「変数（へんすう）」とはデータを一時的に入れておく入れ物のことです（次のレッスンでさらに説明します）。変数にはユーザーが自由に定義できるものや、プロシージャが処理のためにあらかじめ用意しているものなどがあります。

このレッスンで登場した、プロシージャに渡す変数は、正確には「引数（ひきすう）」と呼ばれます。引数はユーザーが勝手に変更することはできません。この例では、「Form_MouseDown」というプロシージャを処理するためにButton、Shift、X、Yという4つの異なる引数が用意されています。

Visual Basic初心者なら、別にどちらも変数だということで、大きくとらえてしまってもかまいません。本書では特に区別する必要がない限り、どちらも変数という呼び方をしています。

4 このプロシージャでは最初に、マウスのどのボタンが押されたのかをチェックします。そして、そのボタンが左ボタン（Button=1）であれば、ラベルコントロール「Label1」の［Left］プロパティにマウスのX座標位置を、［Top］プロパティにマウスのY座標位置を代入します。

```
If Button = 1 Then
    Label1.Left = X
    Label1.Top = Y
End If
```

5 左ボタンが押されたかどうかを判断するのに、Ifステートメントを使用しています。ユーザーが左ボタンを押せば、プロシージャの変数Buttonに「1」が格納されます。この値を使って、もし変数Buttonの値が左ボタンであれば、ラベルコントロールを動作させる処理を実行します。それ以外のボタンが押された場合は、なんの処理も行いません。

6 ラベルコントロールの表示位置を変えるには、［Left］と［Top］プロパティの値を変えます。［Left］は、ラベルコントロールの左右方向、［Top］は上下方向の位置を設定します。ここに、MouseDownイベントが発生したマウスの座標位置を格納している変数「X」と「Y」の値を代入すれば、ボタンが押された位置がラベルコントロールの［Left］と［Top］プロパティに設定されます。

［Left］と［Top］プロパティの値が変わると、ラベルコントロールはその位置に再表示されます。［Left］と［Top］は、フォームの左上を原点（0、0）として、そこからの距離を設定するプロパティです。

プログラムを実行して、フォームの好きな位置をマウスでクリックしてください。ボタンを押した時点で、その位置に「Welcome!」という文字が移動します。

まとめ

Click イベントは、発生してもプロシージャになにも情報を返してこないので、() 内が空白になっています。MouseDown イベントは、いろいろな情報を返してくるので、() 内にどのような情報が返ってくるのかが記述されています。

ゲームプログラムなどで、クリックされた場所を使いたい場合には、この MouseDown イベントプロシージャを使うと便利です。

練習問題

Q1　フォームの上でマウスの右ボタンを押したときに、ラベルコントロールが移動するように変更してください。

【ヒント】

MouseDown イベントプロシージャでは、マウスの右ボタンが押されると、変数 Button には「2」が格納されます。

解答は巻末に→

ワンポイントアドバイス

If ステートメントは、ある値や式の結果で、以降の処理を振り分けたいときなどに使います。式の結果が真（True）の場合は、「End If」までの処理を実行します。

【書式】

```
If 式 Then
        .....処理1
        .....処理2
End If
```

今回の例題では、押されたボタンの種類を使って、処理を振り分けています。

Lesson 10

第２日
３時限目

イベントを使う

Lesson10.vbp
VB6CD
Lesson10

◎Learning Edition

変数を使う

イベントプロシージャで扱う変数の使い方を通じて、「変数」について理解を深めます。

今回作成するプログラム

変数を使う

Label1

1 任意の位置をクリックすると…

変数を使う

マウスの位置は3675,1665

2 クリックした位置の座標が表示されます

POINT

MouseUpイベントを使って、フォームでボタンが離された位置をラベルコントロールに表示します。

MouseUpが発生すると、そのときのマウスに関する情報が、イベントプロシージャの変数に格納されます。この変数を、文字列と組み合わせて、ラベルコントロールで表示します。

コントロールとプロパティの設定

※新しいプロジェクトとフォームを用意します。

1 Form1の［Caption］プロパティの値を「変数を使う」に変更する

(オブジェクト名)	Form1
Appearance	1 - 3D
AutoRedraw	False
BackColor	&H00C0C0C0&
BorderStyle	2 - 可変
Caption	変数を使う
ClipControls	True

1 ここをダブルクリック

2 「変数を使う」と入力

2 ツールボックスの［Label］ A をクリックし、フォーム上でドラッグしてラベルコントロールを配置する

1 ここをクリックして…

2 そのままここまでドラッグ

3 プロパティウィンドウで［BorderStyle］プロパティをクリックし、［▼］をクリックして［1 - 実線］を選ぶ

(オブジェクト名)	Label1
Alignment	0 - 左揃え
Appearance	1 - 3D
AutoSize	False
BackColor	&H8000000F&
BackStyle	1 - 不透明
BorderStyle	1 - 実線
Caption	0 - なし / 1 - 実線
DataField	
DataFormat	
DataMember	

1 ここをクリック

2 ここをクリック

3 ここをクリック

ツールボックスのボタン名

ツールボックスのボタンの上にマウスポインタを移動すると、そのボタンの名前が表示されます。

ラベルを選択

この操作を行う前に、作成したラベルコントロールが選択されていることを確認してください。

プロパティ値の切り替え

［BorderStyle］プロパティをダブルクリックして、［0 - なし］と［1 - 実線］を切り替えることもできます。

コードの記述

Form1:From-MouseUp

Project1 - Form1 (コード)

Form / MouseUp

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, _
                         X As Single, Y As Single)
    Dim Message As String

    If Button = 1 Then
        Message = "マウスの位置は" & X & "," & Y
        Label1.Caption = Message
    End If
End Sub
```

解説

1 MouseUpイベントは、ユーザーがマウスのボタンを離したときに発生するイベントです（押したときはMouseDownイベントでしたね）。このプログラムでは、フォームの上でMouseUpイベントが発生したときの処理を作成しますので、フォームのMouseUpイベントプロシージャを使います。フォームをダブルクリックしてコードウィンドウを開き、イベント用リストボックスからMouseUpを選びます。

Project1 - Form1 (コード)

Form / MouseUp

MouseDown
MouseMove
MouseUp
OLECompleteDrag
OLEDragDrop
OLEDragOver
OLEGiveFeedback
OLESetData
OLEStartDrag
Paint
QueryUnload
Resize

1 ここをクリック

2 ここをクリック

2 MouseDownイベントと同様に、このイベントが発生すると、そのときのマウスボタンの種類、キーの種類、およびマウスポインタの座標が、イベントプロシージャの変数に格納されます。

このプログラムは、マウスの左ボタンが押されて離されたときのマウスポインタのX/Y座標位置を、文字列と組み合わせてメッセージを作成し、ラベルコントロールに表示します。

3 最初に、変数の宣言を記述します。「Message」という変数を使います。

プロシージャに「Dim Message As」と入力してスペースキーを押すと、コード

ウィンドウにリストが表示されます。そのまま「String」と入力するか、リストボックスの「String」が選択状態になったら、スペースキーを押して入力します。

Project1 - Form1 (コード)

Form　MouseUp

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer,
    Dim Message As str
End Sub
```

String
SystemColorConstants
TextBox
Timer
UserControl
UserDocument
Variant

スペースキーを押す

4　続いて、Ifステートメントで、押されたボタンが左ボタンの場合のみ処理を実行するようにします。

```
If Button = 1 Then
      .....
      .....
End If
```

5　メッセージの作成は、文字列連結演算子「&」を使って行います。

メッセージを作成する場合は、直接文字列を指定するときはその文字を " (引用符) で囲みます。変数を指定する場合は、そのまま変数名を記述します。そして、それぞれを「&」でつなげれば、1つの文字列が完成します。

```
Message = "マウスの位置は" & X & "," & Y
```

このプログラムでは、マウスの左ボタンが押されて離されるたびに、変数XとYの値を使ってメッセージを作成します。

6　作成したメッセージは、いったん変数「Message」に格納します（この変数の中身は、MouseUpイベントが発生するたびに更新されます）。そして、ラベルコントロールの［Caption］プロパティに、その変数の値を代入します。これで、マウスのボタンが離された座標を示すメッセージが、ラベルコントロールに表示されます。

7　変数とは、データを一時的に入れておく入れ物のことをいいます。常に内容が変更されるので、毎回値が変化する処理に使用します。変数は、ユーザーが自由に作成することができます。そして、変数を使うときは、Dimステートメントを使って「宣言」する必要があります。すなわち、「これからこういう名前の変数を使うよ」というのを、Visual Basicに伝えるのです。通常、Dimステートメントで変数を宣言する場合は、プロシージャの先頭で行います。

```
Dim Message As String
```

8 変数の宣言時には、使用するデータに合わせたデータ型を指定します。たとえば、文字列を格納する変数を使うのであれば、変数名のあとに「As String」という文字列用のデータ型を付け加えます。整数データの変数であれば「As Integer」を記述します。

データ型の名称（型指定の記述）	メモリ使用量
バイト型 (Byte)	1 バイト
ブール型 (Boolean)	2 バイト
整数型 (Integer)	2 バイト
長整数型 (Long)	4 バイト
単精度浮動小数点数型 (Single)	4 バイト
倍精度浮動小数点数型 (Double)	8 バイト
通貨型 (Currency)	8 バイト
日付型 (Date)	8 バイト
オブジェクト型 (Object)	4 バイト
文字列型 (String) (可変長)	10 バイト + 文字列の長さ
文字列型 (String) (固定長)	文字列の長さ
バリアント型 (Variant) (数値)	16 バイト
バリアント型 (Variant) (文字列)	22 バイト + 文字列の長さ

9 データ型の指定は、それぞれのデータサイズに合ったメモリ領域を確保するために行います。しかし、特になにもデータ型を指定しないと、変数はどのようなデータでも受け入れる「Variant」型になります。

Variant型が不都合なのは、たとえば文字データを格納したときでも、最低22バイトもメモリを消費してしまいます。1文字しか格納しないのにVariant型を指定するのはメモリの無駄遣いになります。メモリを有効に使うためにも、データ型に合った変数の型指定を行うようにしてください。

10 変数は、プログラマが自由に名前を付けてかまいませんが、以下のような一定の規則があります。

①変数名の先頭の文字には、数字と記号は使えません。
②変数名には、ピリオド (.) や型宣言文字と同じ名前は使うことができません。
③変数名は、最大で半角255文字まで使用できます。変数名に全角文字の日本語も使用できます。
④同じプロシージャ内で、同じ変数名を使うことはできません。

変数をコード内で使用する場合は、タイプミスに注意してください。次に示す3つの変数は、それぞれまったく別の変数として扱われます。

```
Dim Angel As Integer
Dim Angle As Integer
Dim Angie As Integer
```

特に、「l」（小文字のエル）と「I」（大文字のアイ）、「O」（大文字のオー）と「0」（数字のゼロ）は、見た目は同じようでも別のデータとして扱われるので注意してください。

まとめ

変数は、この例題プログラムのように、プログラムを実行しないとデータが確定しないような場合に使用すると、たいへん大きな効果を生みます。そして、プログラム作成時はコード内に式だけ記述しておいて、実際の処理は実行時に行うような使い方をします。

プロシージャ内で宣言した変数は、そのプロシージャ内のすべての式で使用できます。また、いくつも変数を使う場合は、プロシージャの先頭に宣言をまとめておくと、使用している変数が一目でわかり便利です。

練習問題

Q1　「Lesson10.vbp」の、マウスポインタの座標位置の表示を、ラベルコントロールではなくメッセージボックスで行うように改修してください。

解答は巻末に→

ワンポイントアドバイス

変数を宣言するには、Dimステートメントを使います。「Dim」の後に変数名を書いて「As」で、データ型とつなげます。データ型を指定することで、変数に格納する情報の種類を定義することができ、プログラムのメモリを有効に使うことができます。

【Dimステートメントの書式】

```
Dim 変数の名前 As データ型
```

Lesson 11

第2日 4時限目

イベントを使う

Lesson11.vbp
VB6CD
Lesson11

◎Learning Edition

MouseDown と MouseUp の組み合わせ

複数のイベントプロシージャ間で1つの変数を共有する方法と、2つのイベントを組み合わせたプロシージャの組み立て法を学びます。

今回作成するプログラム

Form1

1 フォーム上でマウスをクリックしながらドラッグする

Form1

2 マウスボタンを離したときに直線が引かれる

POINT

これまでに学習したMouseDownイベントとMouseUpイベントを組み合わせて、フォームに直線を引きます。フォームに直線を引くには、Lineメソッドを使用します。このメソッドは、線の引き始めと引き終わりに座標を指定しますので、これをMouseDownとMouseUpイベントプロシージャのX/Y座標位置を使って実行します。

コントロールとプロパティの設定

※新しいプロジェクトとフォームを用意します。これ以外に、コントロールとプロパティの設定はありません。Form1のコードウィンドウを開いてください。

コードの記述

Form1:(General)-(Declarations)

Project1 - Form1 (コード)
(General) (Declarations)

```
Dim StartX As Integer
Dim StartY As Integer
```

Form1:Form-MouseDown

Project1 - Form1 (コード)
Form MouseDown

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, _
                           X As Single, Y As Single)
    If Button = 1 Then
        StartX = X
        StartY = Y
    End If
End Sub
```

Form1:Form-MouseUp

Project1 - Form1 (コード)
Form MouseUp

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, _
                         X As Single, Y As Single)
    If Button = 1 Then
        Line (StartX, StartY)-(X, Y)
    End If
End Sub
```

解説

1 フォームをダブルクリックして、コードウィンドウを開きます。リストボックスから、［General］、［Declarations］を選びます。

[General]

Project1 - Form1 (コード)

(General)　(Declarations)

[Declarations]

そして、ここに2つの整数型変数を宣言します。
この宣言は、各プロシージャの外で行われることに注意してください。

```
Dim StartX As Integer
Dim StartY As Integer
```

2 フォームのイベントリストボックスからMouseDownイベントを選び、コードウィンドウにForm_MouseDownイベントプロシージャを挿入します。
Ifステートメントで、押されたボタンが左ボタンの場合の処理を作成します。
MouseDownイベントプロシージャでは、変数「StartX」と「StartY」に、マウスポインタの座標が格納されている変数、「X」と「Y」の値を代入しておきます。

```
If Button = 1 Then
    StartX = X
    StartY = Y
End If
```

3 今度は、フォームのイベントリストボックスからMouseUpイベントを選びます。
プロシージャが挿入されたら、離されたボタンが左ボタンの場合の処理を作成します。

```
Private Sub Form_Mouseup(......)
```

COLUMN ・・・・・

モジュールについて

1つのプログラムを、複数の塊に分割したうちの一塊をモジュールといいます。また、Visual Basicでは、「フォームモジュール」「標準モジュール」「クラスモジュール」など、この言葉をプログラムを作成する単位に使っています。

4 フォームのLineメソッドは、開始位置と終了位置を指定すると、その2点間に直線を引きます。

【Lineメソッドの書式】

Line （ 開始位置のx座標, y座標 ） - （ 終了位置のx座標, y座標 ）

このプログラムでは、マウスのボタンが押された位置が線を引く開始位置に、マウスのボタンが離された位置が終了位置になります。線を引く開始位置は、すでにMouseDownイベントプロシージャで変数「StartX」と「StartY」に格納しています。

終了位置は、MouseUpイベントプロシージャの変数「X」と「Y」を使います。

```
If Button = 1 Then
    Line (StartX, StartY)-(X, Y)
End If
```

5 プログラムを実行してください。フォームの上でマウスの左ボタンを押し、そのままマウスを別の場所に動かしてボタンを離すと、この2点間に直線が引かれます。

6 変数の宣言をプロシージャの外で行うと、この変数はすべてのプロシージャから参照可能な変数（モジュールレベル変数といいます）になります。

今回のプログラムのように、複数のプロシージャで1つの変数を共有したい場合は、フォームや標準モジュールなどのモジュールの宣言セクション（「General」「Declarations」セクション）に、変数を宣言します。

この場合、変数の名前付け規則（P.74参照）などは、プロシージャ内の変数と同じ規則になります。

COLUMN

メソッドについて

コントロールやフォームなど、オブジェクトを操作するために実行する処理のことです。プロパティがオブジェクトの外観や動作設定を行うのに対し、メソッドはオブジェクトを直接動作させます。たとえば、この例で使用したLineメソッドは、指定した座標から座標へ線を描きます。DeleteItemメソッドは、コントロールを削除します。Moveメソッドは、オブジェクトを指定した座標に移動します。メソッドは、オブジェクトごとに用意されており、プロパティと同様どのオブジェクトに対して行うのか、オブジェクト名を指定します。

次の処理は、コマンドボタンを指定したX/Y座標位置に移動するメソッドです。

```
CommandButton1.Move X, Y
```

まとめ

複数のプロシージャやモジュールで変数を共有する場合は、プロシージャ内ではなく、必ず宣言セクションで宣言します。

プログラムが複雑になってくると、いくつものプロシージャで変数を共有して処理を組み立てるようになりますから、変数の使い方は練習して覚えてください。

練習問題

Q1

描画する線を点線に変更してください。線の種類の変更は、コマンドボタンのClickイベントを使います。線の種類を変更するには、フォームの［DrawStyle］プロパティを使用します。

プロパティの値は以下のとおりです

0 －実線（中心線）［既定値］
1 －鎖線
2 －点線
3 －一点鎖線
4 －二点鎖線
5 －透明
6 －実線（ふちどり）

解答は巻末に→

ワンポイントアドバイス

コードウィンドウは、2つの表示方法を切り替えて使用できます。1つは、プロシージャを1つ1つ表示する方法です。もう1つは、すべてのプロシージャを一度に表示する方法です。

これらは、コードウィンドウの左下にあるボタンで切り替えます。いくつものプロシージャをコードウィンドウに記述したときは、必要に応じて表示を切り替えて使用すると、便利です。

```
Private Sub Form_MouseDown(Button As Integer, Shift As Intege
    If Button = 1 Then
        StartX = X
        StartY = Y
    End If
End Sub
```

1つのプロシージャを表示

このボタンをクリック

```
Dim StartX As Integer
Dim StartY As Integer

Private Sub Form_MouseDown(Button As Integer, Shift As Intege
    If Button = 1 Then
        StartX = X
        StartY = Y
    End If
End Sub

Private Sub Form_MouseUp(Button As Integer, Shift As Integer,
    If Button = 1 Then
        Line (StartX, StartY)-(X, Y)
    End If

End Sub
```

複数のプロシージャを表示

プロシージャごとに点線で区切られる

このボタンをクリック

Lesson 12

第２日
５時限目

イベントを使う

Lesson12.vbp
VB6CD
Lesson12

◎Learning Edition

お絵描きプログラムの作成

MouseMoveイベントの使い方を覚えます。また、モジュールレベル変数の使い方を復習します。

今回作成するプログラム

お絵描きプログラム
Color
消去　終了

1 色ボタンをクリックしてからドラッグすると…

お絵描きプログラム
Color
消去　終了

2 カラーの絵を描画できる

3 このボタンで絵を消去

POINT

マウスイベントの１つ「MouseMove」イベントを使って、フォームに自由線を描くプログラムです。フォーム上でボタンを押しながらマウスを動かすと、マウスポインタに沿って線を描画していきます。また、５つのカラーが選択できるカラーパレットを持ち、線の色を変えて描画できます。

コントロールとプロパティの設定

※新しいプロジェクトとフォームを用意します。

1 右下隅のサイズ変更ハンドルをドラッグしてフォームのサイズを広げる

Project1 - Form1 (Form)
Form1

Height:5175、
Width:6105
ぐらいに設定する

1 ここまでドラッグ

2 Form1の［Caption］プロパティの値を「お絵描きプログラム」に変更する

プロパティ - Form1
Form1 Form
全体 | 項目別

AutoRedraw	False
BackColor	&H8000000F&
BorderStyle	2 - 可変
Caption	お絵描きプログラム

1 ここをダブルクリック
2 値を変更

3 ［AutoRedraw］プロパティを［True］に設定する

プロパティ - Form1
Form1 Form
全体 | 項目別

AutoRedraw	True
BackColor	True
BorderStyle	False
Caption	お絵描きプログラム

1 ここをクリック
2 ここをクリック
3 ここをクリック

［AutoRedraw］プロパティ

フォームの［AutoRedraw］プロパティは、フォームがウィンドウの重なりなどでいったん隠れてしまったのち再度表示されたときに、フォームに描画されているグラフィックスを再描画するかどうかを設定するプロパティです。

このプロパティが「False」だと、再描画されず画面が重なった部分は消されたままになってしまいます。

フォームにグラフィックスを描画するときは、必ずこのプロパティを「True」に設定してください。

［True］の設定

［AutoRedraw］プロパティをダブルクリックして、［True］と［False］を切り替えることもできます。

フレームについて

フレームコントロールについては、Lesson7を参照してください。

4 ツールボックスで［Frame］ボタン をクリックし、フォーム上でドラッグしてフレームを配置する

フレームを選択

この操作を行う前に、作成したフレームコントロールが選択されていることを確認してください。

5 プロパティウィンドウで［Caption］プロパティの値を「Color」に変更する

6 ツールボックスの［Label］ボタン をクリックし、フレーム上でドラッグしてラベルを配置する

7 プロパティウィンドウで［BorderStyle］プロパティをクリックし、［▼］をクリックして［1 - 実線］を選ぶ

AutoSize	False
BackColor	&H8000000F&
BackStyle	1 - 不透明
BorderStyle	1 - 実線
Caption	0 - なし
DataField	1 - 実線

1 ここをクリック
2 ここをクリック
3 ここをクリック

8 ［BackColor］プロパティの［▼］ボタンをクリックし、黄色（&H0000FFFF&）に変更する

Appearance	1 - 3D
AutoSize	False
BackColor	&H0000FFFF&

1 ここをクリック
2 ここをクリック
3 ここをクリック
4 ここをクリック

9 ［Caption］プロパティをダブルクリックし、Delキーを押して値を削除する

プロパティー Label1

Appearance	1 - 3D
AutoSize	False
BackColor	&H0000FFFF&
BackStyle	1 - 不透明
BorderStyle	1 - 実線
Caption	
DataField	

1 ここをダブルクリック
2 Delキーを押す

ラベルを選択

この操作を行う前に、作成したラベルコントロールが選択されていることを確認してください。

［BorderStyle］プロパティをダブルクリックして、［0 - なし］と［1 - 実線］を切り替えることもできます。

10 手順6から9を繰り返して青色の四角を作成する

11 同様にして、残りの赤・黄緑・黒の3つのラベルを作成する

青色のコード

「&H00FF0000&」です。

コントロールをコピーする

同じ形のコントロールをたくさん作りたいときは、コントロールをコピーして複製を作ることができます。詳細はP.88のコラムを参照してください。

色のコード

赤：&H000000FF&

黄緑：&H0000FF00&

黒：&H00000000&

12 ツールボックスの［CommandButton］を クリックし、フォームにコマンドボタンを配置する

お絵描きプログラム

Color

Command1

1 ここをクリックして…

2 そのままここまでドラッグ

13 コードウィンドウで［Caption］プロパティの値を「消去」に変更する

プロパティ－ Command1

Command1 CommandButton

全体　項目別

Appearance	1 - 3D
BackColor	&H8000000F&
Cancel	False
Caption	消去
CausesValidation	True

1 ここをダブルクリック

2 「消去」と入力

コマンドボタンを選択

この操作を行う前に、作成したコマンドボタンが選択されていることを確認してください。

14 フォームにコマンドボタンをもう１つ配置する

消去　Command2

1 ここをクリックして…

2 そのままここまでドラッグ

コマンドボタンを選択

この操作を行う前に、作成したコマンドボタンが選択されていることを確認してください。

ユーザーインターフェイス

プログラムの中で、ユーザーと接する部分を表す言葉です。ボタンやパレット、ダイアログボックスなどを総称した呼び方です。

15 プロパティウィンドウで［Caption］プロパティの値を「終了」に変更する

Command2 CommandButton	
(ｵﾌﾞｼﾞｪｸﾄ名)	Command2
Appearance	1 - 3D
BackColor	&H8000000F&
Cancel	False
Caption	終了

1 ここをダブルクリック

2 「終了」と入力

16 完成したユーザーインターフェイス

お絵描きプログラム

Color

消去　終了

COLUMN ･････

コントロールのコピーと貼り付け

コントロールをたくさん作るときに、ドラッグでボタンなどを作っているとサイズが不揃いになることがあります。コントロールを同じ形に揃えたいときは、コントロールをコピーするとよいでしょう。やり方はワープロなどで文字の編集によく使われる「コピー」と「貼り付け」のコンビネーションと同じです。

<コピーする方法>

①コピーしたいコントロールを選択して右クリックから［コピー］を選択。

②貼り付けたいコントロール上で右クリックして［貼り付け］を選択。

③配列にしないので、ダイアログボックスで［いいえ］をクリックする。

④同じコントロールがコピーされる。

コードの記述

Form1:(General)-(Declarations)

Project1 - Form1 (コード)

(General) | (Declarations)

```
Dim CColor As Integer
```

Form1:Command1-Click

Project1 - Form1 (コード)

Command1 | Click

```
Private Sub Command1_Click()
    Form1.Cls
End Sub
```

Form1:Command2-Click

Project1 - Form1 (コード)

Command2 | Click

```
Private Sub Command2_Click()
    End
End Sub
```

Form1:Form-MouseMove

Project1 - Form1 (コード)

Form | MouseMove

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, _
                           X As Single, Y As Single)
    Form1.DrawWidth = 3
    If Button = 1 Then
        PSet (X, Y), QBColor(CColor)
    End If
End Sub
```

Form1:Label1 ～ Label5-Click

Project1 - Form1 (コード)

Label1 | Click

```
Private Sub Label1_Click()
    CColor = 14
End Sub

Private Sub Label2_Click()
    CColor = 0
End Sub
```

```
Private Sub Label3_Click()
    CColor = 9
End Sub

Private Sub Label4_Click()
    CColor = 12
End Sub

Private Sub Label5_Click()
    CColor = 10
End Sub
```

解説

1 フォームをダブルクリックして、コードウィンドウを開きます。コードウィンドウの宣言セクション「General」「Declarations」をリストボックスから選びます。そして、ここに変数「CColor」の宣言を記述します。データ型は整数です。

Project1 - Form1 (コード)
(General) (Declarations)
(General)
Command1
Command2
Form
Frame1
Label1
Label2
Label3
Label4

[General] を選択

```
Dim CColor As Integer
```

2 次に、「Form」の「MouseMove」イベントをリストボックスから選びます。プログラムのメインの処理は、このイベントプロシージャに記述します。

「Form」を選択

Project1 - Form1 (コード)
Form MouseMove

```
Dim CColor As Integer

Private Sub Form_Load()

End Sub
```

Load
LostFocus
MouseDown
MouseMove
MouseUp
OLECompleteDrag
OLEDragDrop
OLEDragOver
OLEGiveFeedback
OLESetData
OLEStartDrag
Paint

ここをクリックし…

ここをクリック

3　MouseMoveイベントは、マウスを動かすと発生するイベントです。ボタンを押さなくてもマウスを動かすだけで発生します。

このイベントが発生すると、MouseMoveイベントプロシージャが呼び出されます。このプロシージャは、実行すると他のマウスイベントプロシージャと同様に、押されたボタンやキーの種類、マウスポインタの座標位置を変数に格納します。

4　このプログラムは、フォームの上でマウスの左ボタンを押しながら動かすと、その座標位置に点を描画します。連続してマウスを動かすと、点は連続して描画され、線になります。

では、MouseMoveイベントプロシージャを順に説明しましょう。

Color

マウスでドラッグしたところに点が描画されていく

5　プロシージャの最初に、描画する点の大きさを設定します。

フォームの［DrawWidth］プロパティは、グラフィック関係のメソッドを使って描画する線の太さを設定します（先に紹介したLineメソッドで線を引く場合も、この［DrawWidth］プロパティを設定すれば線の太さを変えて描画することができます）。

線の太さは、設定する数値が大きくなれば太くなります。最小値は「1」です。ここでは、やや太い「3」を設定します。

```
Form1.DrawWidth = 3
```

6　次に、おなじみになったIfステートメントで、押されたボタンが左ボタンのときだけ、フォームに点を描画する処理を記述します。

フォームに点を描画するには、PSet()メソッドを使います。()内に、描画するX/Y座標を設定すると、その場所にDrawWidthのサイズの点を描画します。また、PSet()のあとに「,」（カンマ）で区切って、描画する点の色を設定できます。

```
If Button = 1 Then
    PSet (X, Y), QBColor(CColor)
```

【Psetメソッドの書式】

```
Pset (描画点のx座標, 描画点のy座標), 描画色の値
```

7 色を設定する場合は、2とおりの方法があります。1つはRGB関数を使う方法です。この方法は、色情報をRed・Green・Blueの3つの値で設定する方法です。

もう1つは、QBColor関数を使う方法です。この関数は、設定されている番号で色を指定する方法で、あらかじめ16色分が番号付けされています。

ここでは、比較的わかりやすいQBColor関数を使います。

```
RGB(red, green, blue)
```

色	Red値	Green値	Blue値
Black	0	0	0
Blue	0	0	255
Green	0	255	0
Cyan	0	255	255
Red	255	0	0
Magenta	255	0	255
Yellow	255	255	0
White	255	255	255

```
QBColor(color)
```

color …色	color …色
0 ・・・・・・・Black	8 ・・・・・・・Gray
1 ・・・・・・・Blue	9 ・・・・・・・Light Blue
2 ・・・・・・・Green	10・・・・・・Light Green
3 ・・・・・・・Cyan	11・・・・・・Light Cyan
4 ・・・・・・・Red	12・・・・・・Light Red
5 ・・・・・・・Magenta	13・・・・・・Light Magenta
6 ・・・・・・・Yellow	14・・・・・・Light Yellow
7 ・・・・・・・White	15・・・・・・Bright White

8 QBColor関数の色番号には、変数「CColor」を指定しておきます。これは、作成したツールパレットから、ユーザーに選ばれた色番号を代入するためです。

```
PSet (X, Y), QBColor(CColor)
```

9

次に、コードウィンドウのリストボックスから「Label1」「Click」を選びます。
フレームに組み込んだラベルコントロールのClickイベントプロシージャに、このQBColor関数に渡す色番号を設定します。たとえば、黄色に塗りつぶしたラベルコントロール「Label1」のClickイベントプロシージャでは、変数CColorに明るい黄色（Light Yellow）の番号「14」を設定します（P.92下表参照）。変数CColorはモジュールレベルの変数として宣言してありますので、MouseMoveイベントプロシージャからこの変数を使うことができます。
これで、ユーザーがこのラベルコントロールをクリックすると、PSetメソッドには明るい黄色が設定されます。そして、マウスをドラッグすると黄色い線がフォームに描画されます。

```
Private Sub Label1_Click()
    CColor = 14
End Sub
```

10

同様に、残りのラベルコントロールのClickイベントプロシージャで、それぞれのラベルコントロールの色と同じ色番号を、変数CColorに代入します。

```
Private Sub Label2_Click()
    CColor = 0   ………黒
End Sub

Private Sub Label3_Click()
    CColor = 9   ………明るい青
End Sub

Private Sub Label4_Click()
    CColor = 12  ………明るい赤
End Sub

Private Sub Label5_Click()
    CColor = 10  ………黄緑
End Sub
```

11

今度は、描画した線を一括消去する処理を作成します。コードウィンドウで、フォームの［消去］ボタン（Command1）のClickイベントプロシージャを開きます。
そして、次の一行を記述します。

```
Form1.Cls
```

Clsメソッドは、実行時に作成したグラフィックスをフォームから消去するメソッドです。このボタンを押すことで、フォームに描画した線を消すことができます。

12 もう1つのコマンドボタン［終了］（Command2）のClickイベントプロシージャに、プログラムの終了処理を記述します。

```
Private Sub Command2_Click()
    End
End Sub
```

13 プログラムを実行してください。マウスの左ボタンを押しながら動かすと、フォームに線が描画されます。ツールパレットの好きな色をクリックすると、今度はその色で線が描画されます。

［消去］ボタンを押せば、何度でも描画することができます。［終了］ボタンを押すとプログラムは終了します。

【Cls メソッドの書式】

`object.Cls`

object	プログラムの実行時に作成したグラフィックスまたはテキストをクリアするオブジェクトを指定します。省略した時は Form オブジェクトが指定されます。

【QBColor 関数の書式】

`QBColor(color)`

color	0～15の範囲の整数（Long型）を指定します（P.92の表参照）。

まとめ

MouseDown、MouseUp、MouseMoveという3つのマウスイベントを勉強しました。グラフィックスを描画するメソッドと組み合わせると、面白いプログラムが作成できます。

また、イメージコントロールと組み合わせてClickイベントを使うと、ゲームプログラムなどが作成できます。

Q 練習問題

Q1 カラーパレットに、「水色（Light Cyan)」を追加し、線の太さを「7」に設定してください。

追加されたカラーパレット

Q2 プログラムに、消しゴムを追加してください。

【ヒント】

①描画した線を消去するには、フォームの背景色と同じ色で線をなぞっていけば、消去したことと同じことになります（ここでは「7（White)」を使っています)。

②消しゴムは、イメージコントロールに消しゴムのアイコン「Erase01.ico」（付属CD-ROMの「Q&A」フォルダに収録）を組み込んで、これをクリックするようにしてください。

消しゴムが付いたお絵描きプログラム

解答は巻末に→

ワンポイントアドバイス

フレームコントロールは、「コンテナ」コントロールとも呼ばれています。これは、フレームの上にラベルやイメージコントロールを配置すると、フレームと一体になって管理することができるからです。フレームにコントロールを配置する場合は、直接フレームに置いてください。一度フォームに配置したコントロールをフレームの上に乗せても、フォームの下に配置されてしまい、うまく表示されなくなってしまいます。このコントロールをフレームに乗せたいときはいったん［切り取り］を実行し、フレームをクリックして選択状態にしてから貼り付けを実行すると、フレーム内に貼り付けることができます。

Color

フレームの中に配置する

Color

フォームから移動しても
フレームに入らない

COLUMN

コメントをつける

コードの働きや機能を説明するときは、ソースコードの中にコメント（説明）をつけることができます。コメントをつけるにはコメント記号「'（シングルクォテーション）」をコメント文の先頭につけます。コメント記号からその行末までの文は、実行時には処理されません。コメントは日本語でも英語でもかまいません。

コメント行はコードウィンドウでは緑色に表示されています。他人にソースコードを配布するときなどは、コメントをつけておくとコードが読みやすくなるでしょう。

```
'コメントは日本語でも入力できます。
    CColor = 0          '番号 0 は黒の指定です。
   'CColor =12          この行のコードは実行されません。
```

Visual Basic 6

Let's VBSchool!!

第3日 (Lesson13～Lesson17)

メニューを使う

Learning School

これまでのレッスンで、だいぶ見栄えのするプログラムが作れるようになりました。

そこで、第3日のレッスンでは、フォームにメニューを組み込んで、さらにアプリケーションらしく仕上げるテクニックを身に付けます。

Windowsのアプリケーションでは、ほとんどがメニューを持っています。

メニューを使うと、プログラムの機能を整理して割り当てておくことができますので、ユーザーにわかりやすいインターフェイスを作成できます。

メニューコントロールは、他のコントロールと少し違った使い方をしますので、この章でメニューコントロールの操作方法を習得してください。

今日のレッスンで解説する項目

- ❏ フォームにメニューを組み込む方法
- ❏ メニュー用のイベントプロシージャを使って、ユーザーがメニューを選択したときに処理を実行する方法
- ❏ ポップアップメニューの作成方法
- ❏ メニューコントロールを追加するテクニック
- ❏ 追加されたメニュー項目が選択された場合の処理方法
- ❏ メニューからメニューコントロールを削除する方法
- ❏ メニューコントロールの［Visible］プロパティを使う方法

Lesson 13

第３日
１時限目

メニューを使う

Lesson13.vbp
VB6CD
Lesson13

◎Learning Edition

メニューを組み込む

フォームにメニューを組み込む方法を習得します。また、ユーザーがメニューを選択したときに処理を実行する方法を覚えます。

今回作成するプログラム

1 ［ファイル］メニューの［新規作成］をクリックすると…

2 絵が消去され新しい絵が描けるようになる

POINT

メニューエディタを使って、Lesson12の練習問題Q2で作成したお絵描きプログラム（Q&A-122.vbp）にメニューを加えます。

２つのコマンドボタンで、別々に処理をしていた「消去」と「終了」の処理を、１つのメニューから実行できるようにします。

コントロールとプロパティの設定

1 「Q&A-122.vbp」を開く

2 [標準] ツールバーの [メニューエディタ] ボタンをクリックする

1 このボタンをクリック

3 [キャプション] に「ファイル」、[名前] に「IDMFile」を入力し、[次へ] をクリックする

1 「ファイル」を入力

2 「IDMFile」を入力

3 [次へ] をクリック

Q&A-122.vbp

「Q&A-122.vbp」ファイルは、付録CD-ROMの「Q&A」フォルダに収録されています。

[ツール] メニューを使う

[ツール] メニューから [メニューエディタ] を選ぶこともできます。

コントロール名

[名前] ボックスに入力した「IDMFile」を、コントロール名と呼びます。コードでメニューコントロールを参照するときに使う名前です。

キャプション

ここにメニュータイトルまたはメニュー項目名を入力すると、そのメニューコントロールがメニューコントロールリストボックスに追加されます。

矢印ボタンの使い方

上下左右の矢印ボタンはメニュー項目を移動させるのに使用します。項目を選んで移動させたい向きの矢印ボタンをクリックします。

4 右矢印ボタンをクリックして名前の配置を右にずらす

5 [キャプション] に「新規作成」、[名前] に「IDMNew」を入力し、[次へ] をクリックする

6 [キャプション] に「-」、[名前] に「separator」を入力し、[次へ] をクリックする

セパレータ(区切り線)

メニューの項目を似たようなカテゴリで区分することでメニューを見やすくします。[キャプション] ボックスに「-」(半角ハイフン) を入力して設定します。

7 [キャプション] に「終了」、[名前] に「IDMExit」を入力し、[OK] をクリックする

8 フォームにメニューが挿入される

実行前はコードウィンドウが開く

プログラムを実行する前にメニューを選択すると、そこに対応したプロシージャのコードウィンドウが開きます。

COLUMN ······

階層メニューを作成するには

メニューエディタで階層メニューを作成する場合は、矢印ボタンを使って右にずらしていきます。通常、メニューは縦に並んで表示されますが、一段下の階層を設定する場合は、１つ右にずらしたメニュー項目を作成します。また、メニューに区切り線を挿入する場合は、必ず［キャプション］ボックスに「-」(半角のハイフン) を入力します。

コードの記述

Form1:IDMNew-Click

```
Project1 - Form1 (ｺｰﾄﾞ)
IDMNew                    Click

Private Sub IDMNew_Click()
    Form1.Cls
End Sub
```

Form1:IDMExit-Click

```
Project1 - Form1 (ｺｰﾄﾞ)
IDMExit                   Click

Private Sub IDMExit_Click()
    End
End Sub
```

解説

すでに、プログラムそのもののコードはLesson12で作成してありますので、追加したメニューのコードだけを解説します。

1 ユーザーがメニューを選択すると、そのメニューのClickイベントが発生します。メニューエディタでコントロール名（名前）を設定しましたが、この名前とClickイベントが組み合わされて、イベントプロシージャになります。

たとえば、ユーザーが［新規作成］メニューを選ぶと、「IDMNew_Click()」というイベントプロシージャが発生します。

2 あとは、コマンドボタンに設定した処理とまったく同じ物を記述すればいいだけです。［新規作成］メニューには、［消去］ボタンで作成した、フォームのグラフィックを消すClsメソッドを、［終了］メニューにはプログラムの終了のためのEndステートメントを記述します。

3 プログラムを実行します。線を描画してメニューから［新規作成］を選びます。［消去］ボタンを押したときと同じ処理が実行されます。［終了］メニューを選ぶとプログラムが終了します。

まとめ

フォームにメニューを組み込む場合は、必ずメニューエディタを使用します。
また、これまで、フォームにラベルやイメージコントロールを配置すると、自動的にコントロール名が付加されていました（Image1やLabel1など）。しかし、メニューコントロールについては、組み込んだメニュー項目に、自分で1つ1つコントロール名（名前）を付けなければなりません。名前は変数のように自由に設定できます。
なお、メニューコントロールは、Clickイベントしか発生しません。他のイベントはありません。

練習問題

Q1 「Lesson13.vbp」に、次の画面のメニューを追加してください。メニュー項目だけ組み込み、処理は記述する必要はありません。

お絵描きプログラム
ファイル　編集
切り取り
コピー(C)
貼りつけ ▶ 貼りつけ
ファイルから
Color

【ヒント】
2段階深い階層のメニューを作成する場合は、右矢印ボタンを2回押します。
また、区切り線を複数挿入する場合は、コントロール名が同じにならないようにしてください。その場合は、コントロール名に「separator1」「separator2」などと番号を振るといいでしょう。

解答は巻末に→

ワンポイントアドバイス

一度作成したメニューを修正する場合は、メニューを組み込んだフォームを表示して、再度Visual Basicのツールバーにある［メニューエディタ］ボタンを押します。
組み込んだメニュー項目を削除したい場合は、メニューエディタでそのメニュー項目を選択して［削除］ボタンを押します。

Lesson 14

第３日
２時限目

メニューを使う

Lesson14.vbp
VB6CD
Lesson14

◎Learning Edition

右ボタンポップアップメニューを作る

ポップアップメニューの作成方法を習得します。マウスをクリックしたときだけ表示するようにしてみましょう。

今回作成するプログラム

お絵描きプログラム
ファイル
新規作成
Color
消去
終了

1 フォーム上で右クリックするとポップアップメニューが表示される

2 メニューをクリックすると…

お絵描きプログラム
ファイル
Color
消去
終了

3 絵が消去され、新しい絵が描けるようになる

POINT

Windowsではおなじみの、マウスの右ボタンでポップアップするメニューを作成します。このポップアップメニューには、PopupMenuメソッドを使います。ただし、実際に表示されるメニューは、あらかじめメニューエディタを使って作成するか、Loadステートメントで実行時にメニュー項目を追加する必要があります。

ここでは、あらかじめ作成してあるメニューを使って、それをポップアップメニューにします。

コントロールとプロパティの設定

※ここでは、付録CD-ROMの「Lesson13」フォルダに収録されている「Lesson13.vbp」ファイルを使用するので、特に設定する必要はありません。

コードの記述

Form1:Form-MouseDown

```
Project1 - Form1 (コード)
Form                MouseDown

Private Sub Form_MouseDown(Button As Integer, Shift As Integer, _
                           X As Single, Y As Single)
    If Button = 2 Then
        IDMExit.Visible = False
        separator.Visible = False
        PopupMenu IDMFile, vbPopupMenuRightButton
        IDMExit.Visible = True
        separator.Visible = True
    End If
End Sub
```

解説

すでに、プログラムそのもののコードはLesson13で作成してありますので、追加するポップアップメニューのコードだけを解説します。

1　ポップアップメニューは、ユーザーがフォームの上でマウスの右ボタンを押している間中表示されます。ですから、この処理はフォームのMouseDownイベントプロシージャに記述します。

2　メニューのポップアップ表示は、PopupMenuメソッドが行います。メソッド名の後にメニューコントロール名を記述すると、そのメニューをポップアップ表示します。

メソッドの引数にメニューの表示位置や、メニューを選択するマウスのボタンを設定できます。

【PopupMenuメソッドの書式】

```
object.PopupMenu menucontrol, flags, x, y, boldcommand
```

object	省略可能。メニューを組み込んであるオブジェクトです。objectを省略すると、フォーカスを持つFormオブジェクトが指定されます。
menucontrol	ポップアップさせるメニュー項目のコントロール名を記述します。指定されたメニューは、最低1つのサブメニューを持っている必要があります。
flags	省略可能。ポップアップするメニューの位置と動作を指定する値（定数）を指定します。
x	省略可能。メニューを表示する位置のx座標を指定します。省略すると、マウスの座標が指定されます。
y	省略可能。メニューを表示する位置のy座標を指定します。省略すると、マウスの座標が指定されます。
boldcommand	省略可能。ポップアップメニューのうち太字で表示されるメニュー名を指定します。

＜Flagsに設定する値＞

定数（位置）	値	内容
vbPopupMenuLeftAlign	0（既定値）	ショートカットメニューの左端がxに配置されます。
vbPopupMenuCenterAlign	4	ショートカットメニューの中央がxに配置されます。
vbPopupMenuRightAlign	8	ショートカットメニューの右端がxに配置されます。
vbPopupMenuLeftButton	0（既定値）	ショートカットメニューの項目は、マウスの左ボタンをクリックしたときにのみ反応します。
vbPopupMenuRightButton	2	ショートカットメニューの項目は、マウスの左または右のボタンをクリックしたときに反応します。

3 このプログラムでは、マウスの右ボタンを押したときにメニューを表示しますので、Ifステートメントを使って、右ボタンが押されたときのみ処理を実行します。

すでに作成してある［ファイル］メニューをポップアップさせますから、このメニュー項目のコントロール名（名前）「IDMFile」をPopupMenuメソッドの第一引数として記述します。

また、メソッドの第二引数に、定数「vbPopupMenuRightButton」を指定します。

この定数をセットすると、表示されたメニューはマウスの左ボタンでも右ボタンでも選択できるようになります。

```
If Button = 2 Then
    PopupMenu IDMFile, vbPopupMenuRightButton
End If
```

4 階層メニューの場合は、上の階層のメニュー名をPopupMenuメソッドに記述すると、その下層のメニュー項目もいっしょに表示されます。

このプログラムでは、［ファイル］メニューをポップアップメニューに指定していますから、その下の［新規作成］［終了］も自動的に表示されます。

5 下層メニューを表示したくない場合は、PopupMenuを実行する前に、そのメニューコントロールの［Visible］プロパティをFalseに設定しておきます。必ずPopupMenuメソッドを実行する前に設定してください。メニューがポップアップされてからでは、表示を取り消せません。このプロシージャでは、「ファイル」メニューの区切り線と［終了］を非表示にして、「新規作成」のみを表示させています。

```
IDMExit.Visible = False ••••••••••••••[終了] メニュー非表示
separator.Visible = False •••••••••••••[区切り線] 非表示
PopupMenu IDMFile, vbPopupMenuRightButton ••••メニュー表示
```

6 しかし、一度サブメニューを非表示にしてしまうと、フォームに組み込んだメニューの区切り線と「終了」メニューも、非表示になったままになってしまいます。これでは、プログラムの終了がメニューから実行できなくなってしまいますので、PopupMenuメソッドを実行した直後に、これらのプロパティに「True」を設定し、表示状態に戻す処理を作成しておく必要があります。

```
IDMExit.Visible = True
separator.Visible = True
```

7 ポップアップメニューは、設定してあるメニューを呼び出すだけですから、すでにメニューコントロールに設定してあるClickイベントプロシージャは、通常のメニューと同様に呼び出され実行されます。

8 以上で完成です。プログラムを実行してください。フォームにポップアップされるメニューは、「新規作成」のみです。

まとめ

ポップアップメニューは、操作の手元にメニューを表示できる、という利点があります。この利点を上手に活用し、ユーザーの操作しやすいインターフェイスを作成するようにしましょう。

練習問題

Q1 Lesson13の練習問題のプログラム（Q&A-13.vbp）で組み込んだ［編集］メニューを、ポップアップメニューにしてください。

【ヒント】
「編集」メニューのコントロール名を、PopupMenuメソッドに指定します。

解答は巻末に→

ワンポイントアドバイス

ポップアップメニューの表示は、なるべくマウスの右ボタンに割り当ててください。左ボタンに割り当てると、他の操作のたびにメニューが表示されてしまいます。

また、Windows本来の右クリックで表示されるポップアップメニューの動作はMouseUpイベント時に発生しているようです。例えば、デスクトップを右クリックしてみて下さい。マウスのボタンを押している間はメニューは表示されません。ボタンを離したときにメニューが表示されています。プログラムに不都合が出なければ、MouseUpイベントにポップアップメニューの動作を記述してもよいでしょう。

COLUMN ……

Enabled(有効)プロパティとVisible(表示)プロパティの違い

Enabledプロパティと Visibleプロパティの違いをここで整理しておきましょう。

まず、Enabledプロパティは、そのフォームやコントロール（メニューやボタン）に対して発生したイベントを受け付けるかどうかを設定するプロパティです。これが偽（False）になっていると、そのオブジェクトのイベントプロシージャは実行されません。つまり、オブジェクトはユーザーにとって完全に使用不可能な状態になっています。

一方、Visibleプロパティは、ただ単にフォームやコントロールを見えなくしているだけです。つまり、内部的には使用可能なので、イベントを受け付ける状態にあります。このレッスンではメニューの表示、非表示を処理してポップアップメニューを実現していましたが、方法によっては、隠しメニューを用意し、それを表示することで、ポップアップメニューを行うことも可能です。

Enabledプロパティを False
Visibleプロパティを True のとき

メニューがグレー表示になりイベントも受け付けない

Enabledプロパティを True
Visibleプロパティを False のとき

メニューが表示されないがイベントは受け付ける

Lesson 15

第3日
3時限目

メニューを使う

Lesson15.vbp
VB6CD
Lesson15

◎Learning Edition

メニューを追加する

メニューコントロールを追加するテクニックを学びます。また、追加されたメニュー項目が選択された場合の処理方法を理解します。

今回作成するプログラム

メニューの追加
ファイル
追加

1 ［追加］ボタンをクリックすると…

メニューの追加
ファイル
開く
上書き保存
Menu1
Menu2
Menu3
追加

2 メニューが追加される

3 メニューをクリックすると…

メニューの追加
ファイル
Menu3が選択されました
追加

4 メッセージが表示される

POINT

メニューコントロールは、プログラムデザイン時に作成するだけではなく、プログラム実行時に自由に追加することができます。

このプログラムでは、コマンドボタン［追加］が押されると、新しいメニューを追加していきます。追加されたメニューを選ぶと、メニュー項目名がラベルコントロールに表示されます。

コントロールとプロパティの設定

※新しいプロジェクトとフォームを用意します。

1 ツールボックスの［Label］ をクリックし、フォーム上でドラッグしてラベルコントロールを配置する

Form1

Label1

1 ここをクリックして…

2 そのままここまでドラッグ

2 プロパティウィンドウで［Caption］プロパティの値を削除する

プロパティ - Label1

Label1 Label

全体 | 項目別

BackColor	&H8000000F&
BackStyle	1 - 不透明
BorderStyle	0 - なし
Caption	
DataField	
DataFormat	
DataMember	
DataSource	

1 ここをダブルクリック

2 Del キーを押して「Label1」を削除

3 ［BorderStyle］プロパティをクリックし、［▼］をクリックして［1 - 実線］を選ぶ

プロパティ - Label1

Label1 Label

全体 | 項目別

BackColor	&H8000000F&
BackStyle	1 - 不透明
BorderStyle	1 - 実線
Caption	0 - なし
DataField	1 - 実線
DataFormat	
DataMember	

1 ここをクリック

2 ここをクリック

3 ここをクリック

フォームのサイズ

図を参考にして、フォームの高さを変更してください。

ラベルを選択する

この操作を行う前に、作成したラベルコントロールが選択されていることを確認してください。

罫線の設定

［BorderStyle］プロパティをダブルクリックして、［0 - なし］と［1 - 実線］を切り替えることもできます。

4 ツールボックスの［CommandButton］ を クリックし、フォームにコマンドボタンを配置する

Command1

1 ここをクリックして…

2 そのままここまで ドラッグ

5 プロパティウィンドウで［Caption］プロパティの値を「追加」に変更する

プロパティ－Command1

Command1 CommandButton

全体 | 項目別

(オブジェクト名)	Command1
Appearance	1 － 3D
BackColor	&H8000000F&
Cancel	False
Caption	追加

1 ここをダブルクリック

2 「追加」と入力

6 ［標準］ツールバーの［メニューエディタ］をクリックし、メニューを追加する

キャプション	名前
ファイル	IDMFile
…開く	IDMOpen
…上書き保存	IDMSave

1 表のデータどおりに入力していく

メニュー エディタ

キャプション(P): 上書き保存　OK

名前(M): IDMSave　キャンセル

インデックス(X):　ショートカット(S): (なし)

ヘルプ コンテキスト番号(H): 0　ネゴシエート位置(O): 0 － なし

チェック(C)　有効(E)　表示(V)　ウィンドウ リスト(W)

← → ↑ ↓ 次へ(N) 挿入(I) 削除(T)

ファイル
····開く
····上書き保存

コマンドボタンを選択する

この操作を行う前に、作成したコマンドボタンコントロールが選択されていることを確認してください。

メニューの追加

メニューを追加する方法については、Lesson13を参照してください。

7 メニューに区切り線を追加し、[インデックス] に「0」を入力して [OK] をクリックする

メニュー エディタ
キャプション(P): -
名前(M): Separator
インデックス(X): 0
ショートカット(S): (なし)
ヘルプ コンテキスト番号(H): 0
ネゴシエート位置(O): 0 - なし
チェック(C)　有効(E)　表示(V)　ウィンドウ リスト(W)
次へ(N)　挿入(I)　削除(T)
OK　キャンセル
ファイル
····開く
····上書き保存
····-

1 「-」を入力
2 「Separator」を入力
3 「0」を入力
4 [OK] をクリック

インデックス番号

メニューコントロール配列の各要素を識別するために使用されます。

コントロール配列

同じ種類のコントロールを配列で生成・管理できるコントロールのことです。

コードの記述

Form1:(General)-(Declarations)

Project1 - Form1 (コード)　(General)　(Declarations)

```
Dim iCount As Integer
```

Form1:Form-Load

Project1 - Form1 (コード)　Form　Load

```
Private Sub Form_Load()
    iCount = 0
End Sub
```

Form1:Command1-Click

Project1 - Form1 (コード)　Command1　Click

```
Private Sub Command1_Click()
    iCount = iCount + 1
    Load Separator(iCount)
    Separator(iCount).Caption = "Menu" & iCount
End Sub
```

Form1:Separator-Click

Project1 - Form1 (コード)

Separator　Click

```
Private Sub Separator_Click(Index As Integer)
    Dim Msg As String

    Msg = Separator(Index).Caption & "が選択されました"
    Label1.Caption = Msg
End Sub
```

解説

1 このプログラムは、コマンドボタン［追加］を押すと、メニューに次々と「Menuxx」というメニューコントロールを追加していきます。また、追加されたメニューコントロールを選択すると、ラベルコントロールにそのメニュー項目名を表示します。メニューコントロールの追加は、コマンドボタンのClickイベントプロシージャで行います。

2 メニューコントロールを追加するには、プログラムデザイン時に、あらかじめなんらかのメニューコントロールを1つ配置しておきます。そのとき、そのコントロールの［インデックス］を「0」に設定しておくことで、このコントロールを「コントロール配列」にしておきます。

このプログラムでは、区切り線「Separator」のインデックスを「0」にセットしました。これで、このSeparatorメニューは「コントロール配列」として扱えるようになります。

3 コントロール配列というのは、同じ種類のコントロールを配列で生成・管理できるコントロールのことをいいます。たとえば、メニューコントロール「Separator」は、コントロール配列にすると、同じ名前のメニューコントロールをいくつも作成することができます。しかし、それぞれのコントロールは「インデックス」(Indexプロパティ）の番号で区別することができます。

コントロール配列の利点は、インデックス番号－すなわち配列の要素数に変数を使えることです。

4 コントロールを配列で設定した場合は、そのコントロールに対するイベントプロシージャには、新たに「Index」という変数が付加されます。この変数Indexの値には、どのコントロールでイベントが発生したのか、というインデックス番号が格納されます。

したがって、コントロール配列1つ1つにイベントプロシージャが設定されて呼び出されるのではなく、イベントプロシージャは1つだけで、コントロールは配列番号で管理されます。

```
Private Sub separator_Click(Index As Integer)
```

5 では、実際にメニューコントロールを追加するプロシージャを作成しましょう。

まず、フォームモジュールの宣言セクション「General」「Declarations」に、配列番号を設定するための変数「iCount」を宣言します。この変数は、フォームのLoadイベントプロシージャで、プログラム起動時に0に初期化しておきます。

```
Dim iCount As Integer

Private Sub Form_Load()
    iCount = 0 ・・・・・・・・・・プログラム起動時の初期化
End Sub
```

6 メニューコントロールを追加するのは、Loadステートメントが行います。このステートメントは、ステートメント名のあとにコントロールの名前を記述します。

【Loadステートメントの書式】

```
Load object
```

object　　ロードするオブジェクト名やコントロール配列の要素名を記述します。

7 このとき、コントロール配列を指定するのであれば、コントロール名のあとに配列番号を指定する必要があります。

最初にメニューエディタで設定したSeparatorは、インデックスを「0」に設定していますので、コントロール名（オブジェクト名）は「Separator(0)」になります。したがって、次に追加するコントロール名は「Separator(1)」になります。

コマンドボタンCommand1のClickイベントプロシージャに、メニューコントロールを追加する処理を記述します。

コマンドボタンが押されるたびに、メニューコントロールが追加されますから、配列番号に変数iCountを指定し、ボタンが押されるとこの変数を1つ増加させます。

これで、コマンドボタンが押されるたびに、Separator(1)、Separator(2)、Separator(3)、と新しくメニューコントロールが追加されていきます。

```
iCount = iCount + 1   ・・・・iCountに1を加算する
Load Separator(iCount)  ・・メニューコントロールSeparatorを追加
```

変数の値	Separator
iCount = 0	Separator(0)
iCount = 1	Separator(1)
iCount = 2	Separator(2)

「Separator(1)」を指定することでインデックス番号iCountの1番目を指定できる

8 メニューコントロールが追加されたら、メニューに表示されるメニュー項目名（キャプション）を設定します。このキャプションは、文字列と現在の配列番号を使って作成します。

キャプションは、コントロールの［Caption］プロパティで設定します。

```
Separator(iCount).Caption = "Menu" & iCount
```

9 追加したメニューが選択されると、メニューコントロールSeparatorのClickイベントが発生しますので、このイベントプロシージャで処理を行います。

Separator_Click()イベントプロシージャは、()内にどのコントロールが選択されたのかを示す、コントロール配列の番号を格納する変数「Index」を持っています。このIndexの値を調べれば、何番目に追加したコントロールなのかがわかります。

この変数Indexを使って、選択されたメニューコントロールの［Caption］プロパティを取り出し、ラベルコントロールで表示します。

```
Private Sub Separator_Click(Index As Integer)
    Dim Msg As String

    Msg = Separator(Index).Caption & "が選択されました"
    Label1.Caption = Msg
End Sub
```

10 プログラムを実行します。［追加］ボタンを1回押すたびに、メニューを見てください。Menu1、Menu2、Menu3、と新しいメニューが増えていきます。

そして、そのメニューを選択すると、どのメニューが選択されたかがラベルコントロールに表示されます。

COLUMN

配列について

配列というのは、同じ種類のデータやオブジェクトを1つにまとめたものです。Visual Basicのコントロールは、ほとんどが配列で配置できます。配列を使うと、変数やコントロールの名前は1つで、その後に「添字」と呼ばれるインデックス番号を付加することで、1つ1つのデータ（またはオブジェクト）を識別できるようになります。

配列の利点は、データを1つの名前と番号で管理・操作できる点です。コントロールも1つの名前と番号で操作できます。そのため、番号に変数を使うことができるようになり、式を使ってプログラム実行時にデータやコントロールにアクセスできるようになります。

まとめ

プログラム実行時にメニューを追加する場合は、デザイン時に必ずダミーのメニューコントロールを１つ作成しておきます。このとき、メニュー項目名が混乱しないように、区切り線をダミーに使うのがポイントです。このダミーのコントロールのインデックスを「0」に設定して、コントロール配列にしておきます。
実際にメニューコントロールを追加するのは、Loadメソッドが行います。

練習問題

Q1　「Lesson15.vbp」を使って、最初に追加されたメニューのキャプションが「追加」に、２番目に追加されたメニューのキャプションが「印刷」になるように、プロシージャを改修してください。また、メニューが２つ追加された時点で、メニューが追加できないように処置してください。

【ヒント】
最初に、コントロール配列のインデックス番号を調べ、Ifステートメントで処理を分岐します。
あとは、コントロール名の配列番号を、変数ではなく直接数値で指定して、[Caption] プロパティにメニュー項目名の文字列を代入します。

解答は巻末に→

ワンポイントアドバイス

設定したメニューを使えないように（グレー表示）するには、メニューコントロールの [Enabled] プロパティを「False」に設定します。これで、メニューに項目名は表示されていますが、グレー表示になって使用できない状態になります。
元に戻すには、[Enabled] プロパティを「True」に設定します。

`IDMOpen.Enabled = False` のとき

メニューの追加
ファイル
開く
上書き保存
メニューがグレー表示になって動作しない

Lesson 16

第３日
４時限目

メニューを使う

Lesson16.vbp
VB6CD
Lesson16

◎Learning Edition

メニューの削除

メニューからメニューコントロールを削除する方法を覚えます。比較演算子を使ってみます。

今回作成するプログラム

メニューの追加
ファイル
開く
上書き保存
Menu1
追加
削除

1 ［削除］をクリックすると…

メニューの追加
ファイル
Menu1 を削除しました
追加
削除

2 メッセージが表示され…

メニューの追加
ファイル
開く
上書き保存
を削除しました
追加
削除

3 「Menu」が削除される

POINT

Microsoft Excelなどでは、セルを操作する場合とグラフを操作する場合では、メニューの項目名が入れ替わる場合があります。

これらは、メニューの追加処理と、メニュー削除の処理を上手に組み合わせて行っています。このレッスンでは、メニューからメニューコントロールを削除するテクニックを覚えます。

コントロールとプロパティの設定

※ここでは、前回の「Lesson15.vbp」ファイルに追加しますので、このプロジェクトファイルを開いてください。

1 ツールボックスの［CommandButton］をクリックし、フォームにコマンドボタンを配置する

1 ここをクリックして…

2 そのままここまでドラッグ

2 プロパティウィンドウで［Caption］プロパティの値を「削除」に変更する

1 ここをダブルクリック

2 「削除」と入力

3 ［(オブジェクト名)］の値を「Del」に変更する

1 ここをダブルクリック

2 「Del」と入力

Lesson15.vbpの保存場所

このファイルは、付録CD-ROMの「Lesson15」フォルダに収録されています。

コマンドボタンを選択する

この操作を行う前に、作成したコマンドボタンコントロールが選択されていることを確認してください。

ボタンを揃える

ボタンなどのコントロールを正確に揃えたいときは［書式］メニューを使いましょう。［書式］メニューは複数のコントロールを選択してから使います。一番最後にクリックしたコントロールのプロパティを基準として書式が整えられるのがポイントです。

［整列］メニューでは、上下左右にコントロール同士を揃えることができます。

［同じサイズに揃える］メニューでは、幅や高さを同じ値で揃えることができます。

4 ボタンをドラッグして配置を調整する

メニューの追加
ファイル
追加　削除

COLUMN ……

書式メニューについて

フォームに配置したコントロールの位置や上下の重なりを調整する場合は、コントロールを直接操作する以外に、Visual Basicの［書式］メニューを使うと便利です。

また、複数のコントロールを選択状態にすると、これらをまとめて移動することができます。［書式］メニューの［コントロールのロック］を使うと、そのコントロールを移動することができなくなります。すでに位置を決めたコントロールは、この機能でロックしておきましょう。

コードの記述

Form1:Del-Click

Project1 - Form1 (コード)　Del　Click

```
Private Sub Del_Click()
    If iCount <> 0 Then
        Unload Separator(iCount)
        Label1.Caption = "Menu" & iCount & "を削除しました"
        iCount = iCount - 1
    Else
        MsgBox "削除できません"
    End If
End Sub
```

解 説

1

［削除］ボタンを押すと、追加したメニューを最後に追加した順番に削除していきます。メニューコントロールを削除する処理は、［削除］ボタンのClickイベントプロシージャで行います。コントロールを削除するには、Unloadステートメントを使用します。

```
Unload object
```

objectに、削除するコントロールのオブジェクト名を記述します。コントロールが配列の場合は、インデックスを付けて指定します。

2

［削除］ボタンのプロシージャでは、最初にIfステートメントを使って、追加したメニューがあるのかないのかを判断します。

変数iCountの値を調べて、その値が「0」でなければ（iCount<>0）、メニューコントロールの削除を行います。削除を実行した後、ラベルコントロールを使ってそのコントロール名とメッセージを表示します。

```
If iCount <> 0 Then
    Unload Separator(iCount)
    Label1.Caption = "Menu" & iCount & "を削除しました"
```

3

Ifステートメントで変数iCountの値を調べるのに、<>という記号を使っています。この記号は、「比較演算子」といい、2つのデータを比較するときに使います。<>の記号は、「等しくない」ことを判断し、2つのデータが等しくなければ「真」を返します。

ここでは、変数iCountの値と数値「0」を比較し、等しくなければこの演算子は「真」を返しますので、Ifステートメントが動作しThen以降の処理が実行されます。

演算子	意味	使い方
<	より小さい	A < B
<=	以下	A <= B
>	より大きい	A > B
>=	以上	A >= B
=	等しい	A = B
<>	等しくない	A <> B

4

メニューコントロールを削除したら、変数iCountの値を1つ減らしておきます。この処理をしておかないと、存在しないメニューコントロール配列の番号が残ってしまい、プログラムが異常動作を起こしてしまいます。

```
iCount = iCount - 1
```

5 変数iCountの値が「0」であれば、メニューの追加はありませんから削除ができません。その場合は、メッセージボックスで「削除できません」と表示します。

```
Else
    MsgBox "削除できません"
```

6 Elseステートメントは、Ifステートメントで使った条件以外の条件のときに使います。最初のIfステートメントでは、変数iCountの値が0でない場合で処理を記述しました。それ以外のiCountの値が0であったときの処理は、このElseステートメントを使います。次の例のIfステートメントは、変数iの値が0の場合は処理1と処理2を実行し、0でない場合は処理3を実行します。

```
If i = 0 Then
    ...処理1   ← 式が真（True）の場合
    ...処理2   ←
Else
    ...処理3   ← 式が偽（False）の場合
End If
```

このように、IfステートメントとElseステートメントを組み合わせると、1つの条件で処理を2つに振り分けることができます。

7 プログラムを実行します。［追加］ボタンを押さないで［削除］ボタンを押すと、「削除できません」というメッセージが表示されます。

COLUMN ……

Ifステートメントの使用上の注意

複数の処理が必要なときは「Then」の後で改行して、最後に「End If」を記述します。また「End If」は、必ず行頭から始まるように記述します。1行だけの処理の時は「Then」の後に続けて書くことができますが、その際「End If」は必要ありません。

【正しい記述】

```
If Check1.Value = 1 Then
    MsgBox "チェックされています"
End If
（又は） If Check1.Value = 1 Then  MsgBox "チェックされています"
```

【エラーになってしまう記述】

```
If Check1.Value = 1 Then  MsgBox "チェックされています"
End If
```

［追加］ボタンでメニューコントロールを追加した後に、［削除］ボタンを押すと、最後に追加されたメニューコントロールが削除されます。

それぞれ、メニューを表示しながら、追加・削除の動作を確認してください。

まとめ

「追加」と「削除」の処理を組み合わせると、プログラムの実行中でも状況に応じたメニュー構成を自由に作成できます。

この場合、メニューが選択された処理は、そのメニューのイベントプロシージャで行いますから、メニューが表示されていても、あらかじめ処理を作成しておく必要がありますので注意してください。

練習問題

Q1　作成したプログラムを使い、追加したメニューの２番目（Menu2）が選択されたときに、ラベルコントロールの文字色を「赤」に変えるようにしてください。

【ヒント】

①処理は、メニューコントロールのClickイベントプロシージャで行います。その際、コントロール配列のインデックスが「2」の場合の処理を作成することになります。特定のコントロール配列を指定する場合は、直接配列番号を指定します。

②インデックスの値を調べるには、Ifステートメントが便利です。

メニューの追加
ファイル
Menu2が選択されました
追加　削除

解答は巻末に→

ワンポイントアドバイス

コントロール配列になっていないメニューをUnloadステートメントで削除してしまうと、そのイベントプロシージャも消滅してしまい、あとから追加できなくなってしまいます。その場合は、Visibleプロパティを使って、メニューの表示・非表示を切り替えるようにします。

Lesson 17

第３日
５時限目

メニューを使う

Lesson17.vbp
VB6CD
Lesson17

◎Learning Edition

状況感知型メニューを作成する

メニューコントロールの［Visible］プロパティを使う方法をマスターします。

今回作成するプログラム

Form1
ファイル　表示
開く
保存
終了

Form1
ファイル　表示
フォームの表示

Form2
ファイル
コピー

1 右クリックでポップアップメニューが表示される

2 ［表示］－［フォームの表示］をクリックすると…

3 Form2 が表示される

4 右クリックでポップアップメニューが表示される

POINT

Windows 95 のアプリケーションでよく使われているポップアップメニューは、状況に応じてメニュー項目が変化します。

配置したメニューコントロールの［Visible］プロパティを使うと、この状況に応じて変化するポップアップメニューを作成できます。

このプログラムは、ポップアップメニューのときだけ専用のメニューを表示するプログラムです。２つのフォームを持ち、それぞれが別のポップアップメニューを表示します。

コントロールとプロパティの設定

※新しいプロジェクトとフォームを用意します。

1 ［標準］ツールバーの［メニューエディタ］ を
クリックする

2 ［メニューエディタ］ダイアログボックスで、次の
メニューを設定する

キャプション (Caption)	名前 (Name)	表示 (Visible)
ファイル	IDMFile	チェック
…開く	IDMOpen	なし
…保存	IDMSave	なし
…-	FileSeparator	なし
…終了	IDMExit	チェック
表示	IDMView	チェック
…フォームの表示	IDMForm	チェック

メニュー エディタ
キャプション(P): フォームの表示　OK
名前(M): IDMForm　キャンセル
インデックス(X): 0　ショートカット(S): (なし)
ヘルプ コンテキスト番号(H): 0　ネゴシエート位置(O): 0 - なし
チェック(C)　有効(E)　表示(V)　ウィンドウ リスト(W)
次へ(N)　挿入(I)　削除(T)
ファイル
····開く
····保存
····-
····終了
表示
····フォームの表示

メニューの設定

メニューを設定する方法については、Lesson13を参照してください。

フォームモジュールの一覧

［フォームモジュールの追加］ボタンの右側にある［▼］をクリックすると、フォームモジュールの一覧が表示されます。

次回からこのダイアログボックスを表示しない

［次回からこのダイアログボックスを表示しない］チェックボックスをクリックしてチェックしておくと、次回からこのダイアログボックスが表示されなくなります。

プロジェクトエクスプローラで確認

フォームモジュールを追加するとプロジェクトエクスプローラで確認することができます。

3 ［標準］ツールバーの［フォームモジュールの追加］をクリックし、［フォームモジュール］を選ぶ

1 ここをクリック

2 ここをクリック

4 ［フォームモジュール］を選択し、［開く］をクリックする

1 ここをクリック

2 ここをクリック

5 Form2が追加される

6 Form2にも、以下のメニューを設定する

キャプション (Caption)	名前 (Name)	表示 (Visible)	有効 (Enabled)
ファイル	IDMFile	チェック	チェック
…終了	IDMExit	チェック	チェック
編集	IDMEdit	なし	なし
…コピー	IDMCopy	チェック	チェック

7 次のようなフォームが完成する

メニューの設定

メニューを設定する方法については、Lesson13を参照してください。

ポップアップ用のコントロール

［編集］メニューだけ、［有効］と［表示］のチェックをはずしてください。

コードの記述

Form1:Form-MouseDown

Project1 - Form1 (ｺｰﾄﾞ)

Form | MouseDown

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, _
                           X As Single, Y As Single)
    If Button = 2 Then
        FileSeparator.Visible = True
        IDMOpen.Visible = True
        IDMSave.Visible = True
        PopupMenu IDMFile, vbPopupMenuRightButton
        IDMOpen.Visible = False
        IDMSave.Visible = False
        FileSeparator.Visible = False
    End If
End Sub
```

Form1:IDMExit-Click

Project1 - Form1 (ｺｰﾄﾞ)

IDMExit | Click

```
Private Sub IDMExit_Click()
    End
End Sub
```

Form1:IDMForm-Click

Project1 - Form1 (ｺｰﾄﾞ)

IDMForm | Click

```
Private Sub IDMForm_Click(Index As Integer)
    Form2.Show 1
End Sub
```

Form1:IDMOpen-Click

Project1 - Form1 (ｺｰﾄﾞ)

IDMOpen | Click

```
Private Sub IDMOpen_Click()
    MsgBox "メニューが選択されました"
End Sub
```

Form1:IDMSave-Click

```
Project1 - Form1 (コード)
IDMSave    Click
Private Sub IDMSave_Click()
    MsgBox "メニューが選択されました"
End Sub
```

Form2:Form-MouseDown

```
Project1 - Form2 (コード)
Form    MouseDown
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, _
X As Single, Y As Single)
    If Button = 2 Then
        IDMEdit.Enabled = True
        PopupMenu IDMEdit, vbPopupMenuRightButton
        IDMEdit.Enabled = False
    End If
End Sub
```

Form2:IDMCopy-Click

```
Project1 - Form2 (コード)
IDMCopy    Click
Private Sub IDMCopy_Click()
    MsgBox "メニューが選択されました"
End Sub
```

Form2:IDMExit-Click

```
Project1 - Form2 (コード)
IDMExit    Click
Private Sub IDMExit_Click()
    Unload Form2
End Sub
```

ワンポイントアドバイス

コードウィンドウでのショートカット

- Ctrl+F2 ‥‥オブジェクトリストボックスの選択
- Tab ‥‥‥‥インデント、またはプロジェクトリスト間を移動
- Ctrl+I ‥‥‥‥クイックヒント表示
- Ctrl+Shift+I‥‥パラメータヒント表示
- Ctrl+J‥‥‥‥プロパティ／メソッドの一覧
- Ctrl+Shift+J ‥‥定数の一覧

解 説

1 このプログラムは、ちょっと複雑な動きをします。

①プログラム起動時は、Form1だけが表示されます。このフォームには、［ファイル］と［表示］の2つのメニューが組み込まれています。

②Form1の上でマウスの右ボタンを押すと、メニューがポップアップします。このメニューには、フォームの［ファイル］メニューに加えて、［開く］と［保存］メニューが追加されています。

③［表示］メニューから［フォームの表示］を選ぶと、もう1つのフォーム「Form2」が表示されます。このフォームは「モーダル」（P.133参照）で起動するため、このフォームを閉じない限りForm1の操作ができません。

④このForm2には、［ファイル］メニューだけが表示されています。しかし、フォームの上でマウスの右ボタンをクリックすると、［コピー］メニューがポップアップされます。

2 フォームのメニューとポップアップメニューは、すべてのメニューをあらかじめメニューエディタで作成しておきます。

ただし、最初から表示しておくメニューと表示しないメニューを、メニューエディタの［表示］チェックボックスで設定しておきます。ポップアップメニューに使うメニューは、［表示］チェックボックスのチェックをはずしておきます。こうしておくと、プログラム実行時は［表示］チェックボックスにチェックを付けたメニューのみが表示されます。

3 では、まずプログラム起動時に表示されるForm1から説明をしていきましょう。

Form1のポップアップメニューは、MouseDownイベントプロシージャを使います。最初に、Ifステートメントを使ってボタンの種類を把握し、右ボタンが押されたときのみ、処理を実行するようにします。このときに行う処理は、プログラム作成時にメニューエディタで［表示］のチェックをはずした、3つのメニューコントロールを［表示］に設定することです。［表示］のプロパティは［Visible］ですから、各コントロールの［Visible］プロパティの値を「True」に設定すると、これらのメニューコントロールは表示状態になります。

```
If Button = 2 Then
    FileSeparator.Visible = True
    IDMOpen.Visible = True
    IDMSave.Visible = True
```

4 メニューを表示状態にしたら、PopUpMenuメソッドを使ってメニューを表示します。このとき、最初の階層のメニューコントロール名（オブジェクト名）を指定すると、次の層のメニューが表示されます。

```
PopupMenu IDMFile, vbPopupMenuRightButton
```

ここでは、「IDMFile」（ファイル）メニューをPopupMenuに指定すると、［ファイル］メニューの下のメニューがポップアップ表示されます。最初の階層のメニュー（メニュータイトル）は、ポップアップメニューでは表示されません。

5　メニューがポップアップされて処理を終えたら、表示状態にしたメニューを、再度非表示にします。こうしておかないと、フォームのメニューにこれらのメニューが表示されてしまいます。

```
IDMOpen.Visible = False
IDMSave.Visible = False
FileSeparator.Visible = False
```

Form1
ファイル　表示
開く
保存
終了

この３つの項目が非表示の設定になっているため［終了］だけが見えている

Form1
ファイル　表示
終了

6　メニューの処理が作成できたら、各メニューのイベントプロシージャを作成します。

まず、［ファイル］メニューから［終了］が選ばれたら、プログラムを終了するので、このイベントプロシージャにEndステートメントを記述します。

```
Private Sub IDMExit_Click()
    End
End Sub
```

7　［開く］と［保存］メニューは、ここでは実際に動作する部分は作成しませんから、メッセージボックスの表示で済ませています。

```
Private Sub IDMOpen_Click()
    MsgBox "メニューが選択されました"
End Sub

Private Sub IDMSave_Click()
    MsgBox "メニューが選択されました"
End Sub
```

8

［表示］メニューの［フォームの表示］を選ぶと、Form2を表示させます。フォームを表示するには、Showメソッドを使用します。

【Showメソッドの書式】

```
object.Show style
```

object	表示するオブジェクト名を記述します。
style	表示するフォームを、「モードレス」で表示するか「モーダル」で表示するかを設定します。

「モードレス」表示とは、呼び出した側も呼び出された側も、どちらのフォームも表示後の操作ができるモードです。一方、「モーダル」表示は、呼び出されたフォームしか操作できず、このフォームを閉じなければ呼び出した側のフォームの操作が一切できないモードです。

このstyleに、「0」を設定する（あるいは省略する）と「モードレス」に、「1」を設定すると「モーダル」表示になります。ここでは、「モーダル」表示でForm2を表示させます。

```
Private Sub IDMForm_Click(Index As Integer)
    Form2.Show 1
End Sub
```

9

続いて、メニューから呼び出されるForm2のプロシージャを作成します。Form2をダブルクリックして、Form2のコードウィンドウを表示してください。2つのフォームで、同じような処理を作成しますから、フォームを間違えないようにしましょう。まず、Form2の上でマウスの右ボタンをクリックすると、ポップアップメニューが表示されます。この処理は、Form2のMouseDownイベントプロシージャで行います。

10

Form2の［編集］メニューは、メニュー作成時にメニューエディタで［有効］(Enabled）チェックボックスのチェックをはずしたので、このメニューは「無効」(使用不可）状態になっています。このため、プログラム実行時は（フォーム表示時は）メニューには表示されません。

そこで、ポップアップメニューが表示される前に、この「編集」メニューを有効(使用可能）状態にしておきます。

```
If Button = 2 Then
    IDMEdit.Enabled = True
```

11

［編集］メニューの使用状態を有効にしたら、PopupMenuメソッドを使って［編集］メニューを表示させます。メニューがポップアップしたら、再び［編集］メニューを使用不可状態に設定します。

これで、フォームのメニューにないメニュー項目を、ポップアップメニューで表示することができます。

```
PopupMenu IDMEdit, vbPopupMenuRightButton
IDMEdit.Enabled = False
```

12

メニューの表示が完成したら、メニューに対応したイベントプロシージャを作成します。

［コピー］メニューが選択されたときの処理は、ここではメッセージの表示にとどめています。

```
Private Sub IDMCopy_Click()
    MsgBox "メニューが選択されました"
End Sub
```

13

Form2の［終了］メニューは、Form2自身を終了させて、操作をForm1に戻す必要があります。ですからEndステートメントでプログラムを終了させるのではなく、Unloadステートメントでフォームをメモリから削除（解放）することで、Form2を終了させます。

プログラム自身の終了は、Form2を閉じたあとで、Form1の［終了］メニューで行います。

```
Private Sub IDMExit_Click()
    Unload Form2
End Sub
```

【Unloadステートメントの書式】

```
Unload object
```

object	アンロードするオブジェクト名またはコントロール名

まとめ

メニューの設定・操作は、あくまでもユーザーが使いやすい形で作成するのが、一番でしょう。わかりやすいユーザーインターフェイスと操作性は、プログラムの質を決めるといっても過言ではありません。

また、Show メソッドを使ったフォームの表示は、ダイアログボックスの表示でよく使うテクニックです。

練習問題

Q1　「Lesson17.vbp」のプログラムのポップアップメニューに、Form2 の表示用メニュー項目「このプログラムについて」を追加して、Form2 を表示するようにしてください。

Q2　Form2 を、イメージコントロールやラベルコントロールを使って、きれいなダイアログボックスに仕上げてください。

また、メニューはすべて削除し、[OK] ボタンを作成してこのボタンで Form2 を閉じるようにしてください。

Form1
ファイル　表示
開く
保存
このプログラムについて
終了
Form2
Visual Basic 6.0
サンプル プログラム
Haruka Seto. 1998.
OK

解答は巻末に→

ワンポイントアドバイス

PopupMenu で表示できるのは、1つのトップレベルメニューだけです。ですから、いろいろなメニューを組み込みたい場合は、Form2 の [編集] メニューのように、ポップアップメニュー専用のメニューを作成しておき、フォームでは非表示にしておいて、ポップアップ時に表示させるようにするといいでしょう。

COLUMN ・・・・・

条件分岐について

条件によって、プログラムの処理を分けるには、Ifステートメントを使うのが最もわかりやすく簡単です。Ifステートメントには以下の3種類があります。

①If...Thenステートメント

```
If 条件1 Then
    処理1
End If
```

条件1が真ならば処理1が実行されます。簡単な条件で分岐させるときに使用します。

②If...Then...Elseステートメント

```
If 条件1 Then
    処理1
Else
    処理2
End If
```

条件1が真ならば処理1が実行されます。偽ならば処理2が実行されます。2つの条件で分岐させるときに使用します。

③If...Then...ElseIfステートメント

```
If 条件1 Then
    処理1
ElseIf 条件2 Then
    処理2
ElseIf 条件3 Then
    処理3
Else
    処理4
End If
```

3つ以上の条件で分岐させたいときは、ElseIfステートメントを使用します。ElseIfの後に条件式を追加することができます。

ただし、3つ以上の多い分岐になってくると、Ifステートメントを使うより、Select Caseステートメントを使う方が自然です。Select CaseステートメントはLesson20で解説しています。

Let's VBSchool!!

第4日 (Lesson18～Lesson21)

コントロールの連携と動作

Learning School

これまでのレッスンで、コントロールとプロパティの使い方をマスターしました。

今度は、更にいろいろなコントロールを使い、またコントロール同士でプロパティを操作しながら、プログラムを組み立てていくテクニックをマスターしましょう。

ほとんどの場合、プログラムを作成するときは複数のコントロールを組み合わせてユーザーインターフェイスを作成します。また、いくつかのコントロールは、データの入力操作を軽減してくれます。

ここでは、主にアニメーションを表示するプログラムを例に取り、これらのコントロールの使い方を説明していきます。

今日のレッスンで解説する項目

- ❏ タイマーコントロールを使ったアニメーションのテクニック
- ❏ グラフィック付きのコマンドボタンでツールバーを作成
- ❏ コントロールとフォームを組み合わせて使うテクニック
- ❏ スクロールバーの使い方
- ❏ テキストボックスコントロールの使い方
- ❏ ツールチップをコントロールに設定する方法
- ❏ チェックボックスコントロールの使い方
- ❏ リストボックスコントロールの使い方
- ❏ リストボックスにリスト項目を追加したり削除するテクニック

Lesson 18

第４日
１時限目

コントロールの連携と動作

Lesson18.vbp
VB6CD
Lesson18

◎Learning Edition

アニメーション表示プログラム

タイマーコントロールを使ったアニメーションのテクニックを覚えます。また、グラフィック付きのコマンドボタンでツールバーを作成します。

●今回作成するプログラム

アニメーション
ファイル(F) 速度
設定
アニメーションプログラム

1 ３つのボタンをクリックすると…
2 アニメーションが一時停止、開始、終了する
3 ［設定］をクリックすると…

表示速度の設定
速度
50ms
100ms
300ms
閉じる

4 速度を指定するダイアログが表示される

POINT

　このプログラムは、タイマーコントロールを使って、イメージコントロールに複数のビットマップを表示し、アニメーション効果を出すプログラムです。
　また、コマンドボタンをツールバー風に仕上げ、ダイアログボックスを使ってアニメーションの表示速度を変更できるようにします。

コントロールとプロパティの設定

※新しいプロジェクトとフォームを用意します。付録CD-ROMの「Lesson18」フォルダに収録されているすべてのアイコンファイル（.ico）とビットマップファイル（.bmp）を、使用しているプロジェクトと同じフォルダにコピーしてください。

1 プロパティウィンドウで［Caption］プロパティの値を「アニメーション」に変更する

プロパティ - Form1
Form1 Form
全体 | 項目別

(オブジェクト名)	Form1
Appearance	1 - 3D
AutoRedraw	False
BackColor	&H8000000F&
BorderStyle	2 - 可変
Caption	アニメーション

1 ここをダブルクリック
2 ここをクリック

2 ［標準］ツールバーの［メニューエディタ］をクリックし、メニューエディタで次のメニューを設定する

キャプション	名前
ファイル(&F)	IDMFile
…終了(&X)	IDMExit
速度	IDMSpeed
…設定...	IDMSet

メニュー エディタ
キャプション(P): 設定
名前(M): IDMSet
OK
キャンセル
インデックス(X):
ショートカット(S): (なし)
ヘルプ コンテキスト番号(H): 0
ネゴシエート位置(O): 0 - なし
チェック(C)　有効(E)　表示(V)　ウィンドウ リスト(W)
次へ(N)　挿入(I)　削除(T)
ファイル(&F)
····終了(&X)
速度
····設定

メニューの設定

メニューを設定する方法については、Lesson13を参照してください。

「設定...」メニュー

メニュー項目名の最後にある「...」の記号は、そのメニューを選択すると何らかのダイアログが表示されるユーザーに示すものです。

3 ツールボックスの［Line］をクリックし、ラインコントロールをフォーム上のメニューの下に配置する

1 ここをクリック

2 ここをクリックして…

3 そのままここまでドラッグ

直線を引く

Shiftキーを押しながらドラッグすると、直線を引くことができます。

4 プロパティウィンドウで［BorderColor］プロパティの色を［ボタンの影］に変更する

1 ここをクリック

2 ここをクリック

3 ここをクリック

4 ここをクリック

ラインを選択

この操作を行う前に、作成したラインコントロールが選択されていることを確認してください。

5 ラインが選択された状態で、［標準］ツールバーの［コピー］と［貼り付け］ボタンを順にクリックする

1 ［コピー］をクリック
2 ［貼り付け］をクリック

6 「コントロール配列にしますか？」というメッセージが表示されるので［はい］をクリックする

1 ［はい］をクリック

7 ラインコントロールが追加される

8 プロパティウィンドウで［BorderColor］プロパティの色を［ボタンの強調表示］に変更する

1 ここをクリック
2 ここをクリック
3 ここをクリック
4 ここをクリック

コントロール配列

同じ名前と同じデータ型を持つコントロールの集合です。コントロールを追加するときにコントロール配列を使うと、複数のコントロールをフォームに追加するときに比べて、リソースの消費を抑えることができます。

ラインの目安

ラインの配置がうまく行かないときは以下の数値を参考にプロパティウィンドウで設定します。

左から→X1/X2/Y1/Y2
Line0・・0/5100/0/0
Line1・・0/5100/15/15
Line2・・0/5100/650/650
Line3・・0/5100/665/665

コントロールを細かく移動する

［ツール］メニューの［オプション］を選択し、［全般］タブで［グリッドに合わせる］チェックボックスをオフにします。

オプション
編集 | エディタの設定 | 全般 | ドッキング | 環境
グリッドの設定
□ グリッドの表示(G)
グリッド単位: twip
幅(W): 120
高さ(H): 120
□ グリッドに合わせる(Q)
☑ ツール ヒントの表示(T)

方向キーでコントロールを移動

コントロールを選択し、Ctrlキーを押しながら方向キー（→、←、↑、↓）を押すと、グリッド間隔に沿って上下左右に少しづつ移動させることができます。

ラインの複製

ラインをコピーするには、手順5と同じように［標準］ツールバーの［コピー］ボタンと［貼り付け］ボタンを使います。

9 ［ボタンの強調表示］の色に設定したラインを下方向に移動する

アニメーション
ファイル(F)　速度

1 わずかにドラッグ

10 2つのラインを囲むようにドラッグして選択する

アニメーション
ファイル(F)　速度

1 ここをクリックして…

2 そのままここまでドラッグ

11 選択したラインを複製し、下方向に移動する

アニメーション
ファイル(F)　速度

1 ここまでドラッグ

12 ツールボックスの［CommandButton］ をクリックし、フォーム上にコマンドボタンを配置する

アニメーション
ファイル(F)　速度
Command1

1 ここをクリックして…

2 そのままここまでドラッグ

13 プロパティウィンドウで［Caption］プロパティの値を削除する

1 ここをダブルクリック

2 Delキーを押す

14 ［Style］プロパティの値を［1 - グラフィックス］に変更する

1 ここをクリック

2 ここをクリック

3 ここをクリック

15 ［Picture］プロパティに「Stop.ico」ファイルを設定する

1 ここをクリック

2 ここをクリック

3 「Stop.ico」を選択

4 ［開く］をクリック

アイコンが表示される

コマンドボタンを選択

この操作を行う前に、作成したコマンドボタンコントロールが選択されていることを確認してください。

値の切り替え

［Style］プロパティのセルをダブルクリックして、［0 − 標準］と［1 − グラフィックス］を切り替えることもできます。

「Stop.ico」の保存場所

［ファイルの場所］ボックスで、このレッスンをはじめる前にアイコンファイルをコピーしたフォルダを指定してください。

16 [(オブジェクト名)] プロパティの値を「StopBtn」に変更する

1 ここをダブルクリック

2 「StopBtn」と入力

17 手順12～16を繰り返し、次の設定でボタンを2つ追加する

設定するプロパティ	スタートボタン (Command2)	終了ボタン (Command3)
(オブジェクト名)	StartBtn	ExitBtn
Caption	なし	なし
Style	グラフィックス	グラフィックス
Picture	start.ico	exit.ico

スタートボタン

終了ボタン

作成したボタンを揃えるには

ボタンの大きさや位置を揃えたいときは［書式］メニューの［整列］や［同じサイズに揃える］を使います。

18 ツールボックスの［Image］ をクリックし、フォーム上にイメージコントロールを配置する

1 ここをクリックして…

2 そのままここまでドラッグ

19 プロパティウィンドウで［Picture］プロパティに「Anime1.bmp」を設定する

アニメーション
ファイル(F)　速度

設定した画像が表示される

20 ツールボックスの［Label］ A をクリックし、フォーム上にラベルコントロールを配置する

アニメーション
ファイル(F)　速度
Label1

1 ここをクリックして…
2 そのままここまでドラッグ

21 プロパティウィンドウで［BorderStyle］プロパティの値を［1 - 実線］に設定する

プロパティ - Label1
Label1 Label
全体　項目別

BackColor	&H8000000F&
BackStyle	1 - 不透明
BorderStyle	1 - 実線
Caption	Label1
DataField	
DataFormat	
DataMember	
DataSource	

1 ここをクリック
2 ここをクリック
3 ここをクリック

Anime1.bmpの設定

［Picture］プロパティを設定する方法については、手順15を参考にしてください。

ラベルを選択

この操作を行う前に、作成したラベルコントロールが選択されていることを確認してください。

［BorderStyle］プロパティをダブルクリックして、［0 - なし］と［1 - 実線］を切り替えることもできます。

22 [Caption] プロパティの値を「アニメーションプログラム」に変更する

23 ツールボックスの [Timer] をクリックし、フォーム上にタイマーコントロールを配置する

タイマーコントロールのサイズ

タイマーコントロールのサイズはあらかじめ決まっています。大きくドラッグしてもタイマーのサイズは変わりません。

24 新しいフォームを追加し、[Caption] プロパティの値を「表示速度の設定」に変更する

フォームの追加

フォームを追加するには、[標準] ツールバーの [フォームモジュールの追加] ボタンを使います。詳しい手順については、Lesson17の手順3と4を参照してください。

25 ツールボックスの［Frame］を使ってフレームを追加し、［Caption］プロパティの値を変更する

表示速度の設定

速度

1 ドラッグしてフレームを作成

2 ［Caption］を「速度」に変更

26 ツールボックスの［OptionButton］を使って、フォーム上にオプションボタンを３つ配置する

表示速度の設定

速度

Option1

Option2

Option3

1 ドラッグしてオプションボタンを作成

27 各オプションボタンの［Caption］と［(オブジェクト名)］プロパティを次の値に変更する

コントロール	Caption	オブジェクト名
Option1	50ms	Option50
Option2	100ms	Option100
Option3	300ms	Option300

表示速度の設定

速度

50ms

100ms

300ms

28 ツールボックスの［CommandButton］をクリックし、フォーム上にコマンドボタンを配置する

100ms
300ms
Command1
1 ここをクリックして…
2 そのままここまでドラッグ

29 プロパティウィンドウで［Caption］プロパティの値を「閉じる」に変更する

100ms
300ms
閉じる
値を変更したボタン

30 完成したレイアウト

アニメーション
ファイル(F) 速度
アニメーションプログラム
表示速度の設定
速度
50ms
100ms
300ms
閉じる

コマンドボタンを選択

この操作を行う前に、作成したコマンドボタンコントロールが選択されていることを確認してください。

中央に表示する

［フォームレイアウト］ウィンドウでForm2の表示位置を［オーナーフォームの中央］に設定すると、オーナーフォームであるForm1に対して中央付近に表示されます。このように表示させるとよりダイアログボックスらしくなります。

フォーム レイアウト
解像度のガイド(G)
開始位置(S)
ドッキング可能(K)
非表示(H)
手動(M)
オーナー フォームの中央(O)
画面の中央(S)
Windows の既定値(W)

コードの記述

Form1:(General)-(Declarations)

Project1 - Form1 (コード)

(General) | (Declarations)

```
Dim ImgNo As Integer
```

Form1:ExitBtn-Click

Project1 - Form1 (コード)

ExitBtn | Click

```
Private Sub Exit_Click()
    End
End Sub
```

Form1:Form-Lord

Project1 - Form1 (コード)

Form | Load

```
Private Sub Form_Load()
    ImgNo = 1
    Timer1.Interval = 100
    StartBtn.Enabled = False
End Sub
```

Form1:IDMExit-Click

Project1 - Form1 (コード)

IDMExit | Click

```
Private Sub IDMExit_Click()
    End
End Sub
```

Form1:IDMSet-Click

Project1 - Form1 (コード)

IDMSet | Click

```
Private Sub IDMSet_Click()
    Form2.Show 1
End Sub
```

Form1:StartBtn-Click

Project1 - Form1 (コード)

StartBtn | Click

```
Private Sub StartBtn_Click()
    Timer1.Enabled = True
    StartBtn.Enabled = False
    StopBtn.Enabled = True
End Sub
```

Form1:StopBtn-Click

Project1 - Form1 (コード)

StopBtn | Click

```
Private Sub StopBtn_Click()
    Timer1.Enabled = False
    StartBtn.Enabled = True
    StopBtn.Enabled = False
End Sub
```

Form1:Timer1-Timer

Project1 - Form1 (コード)

Timer1 | Timer

```
Private Sub Timer1_Timer()
    Dim BMP As String

    BMP = App.Path & "¥anime" & ImgNo & ".bmp"

    If ImgNo = 20 Then
        ImgNo = 0
    End If

    Image1.Picture = LoadPicture(BMP)
    ImgNo = ImgNo + 1
End Sub
```

Form2:(General)-(Declarations)

Project1 - Form2 (コード)

(General) | (Declarations)

```
Dim Speed As Integer
```

Form2:Command1-Click

Project1 - Form2 (ｺｰﾄﾞ)

Command1 | Click

```
Private Sub Command1_Click()
    Form1.Timer1.Interval = Speed
    Unload Form2
End Sub
```

Form2:Form-Load

Project1 - Form2 (ｺｰﾄﾞ)

Form | Load

```
Private Sub Form_Load()
    Speed = 100
End Sub
```

Form2:Option100-Click

Project1 - Form2 (ｺｰﾄﾞ)

Option100 | Click

```
Private Sub Option100_Click()
    Speed = 100
End Sub
```

Form2:Option300-Click

Project1 - Form2 (ｺｰﾄﾞ)

Option300 | Click

```
Private Sub Option300_Click()
    Speed = 300
End Sub
```

Form2:Option50-Click

Project1 - Form2 (ｺｰﾄﾞ)

Option50 | Click

```
Private Sub Option50_Click()
    Speed = 50
End Sub
```

解 説

1 コードの解説の前に、タイマーコントロールについて若干説明をしておきます。

タイマーコントロールは、設定した時間間隔でイベントを発生するコントロールです。[Interval] プロパティに、ミリ秒単位で設定した時間がくるたびに、「Timer」というイベントを発生します。たとえば、[Interval] プロパティに「100ms」を設定すると、0.1秒ごとにTimerイベントが発生します。このTimerイベントプロシージャに処理を記述しておけば、0.1秒ごとにその処理が実行されます。

タイマーコントロールは、ラベルコントロールなどと違い、フォームに配置しても、プログラム実行時は表示されません。

[Interval] プロパティが100msのとき

プログラムの実行開始
100ms後
「Timer」イベント発生 → 「Timer」プロシージャの実行
さらに100ms後
「Timer」イベント発生 → 「Timer」プロシージャの実行
以下プログラム終了まで継続

2 このプログラムは、タイマーコントロールを使って、フォームに配置したイメージコントロールで表示するイメージを、100msごとに入れ替えることによって、アニメーション効果を作るプログラムです。アニメーションは、合計20枚のビットマップファイルを連続表示することで行います。そして、コマンドボタンを使って、このイメージのアニメーションを停止させたり再開させたりします。また、ダイアログボックスを使って、アニメーションの表示速度を変更します。

3　タイマーコントロールを使う場合は、最初に［Interval］プロパティにタイマーコントロールの動作間隔を設定します。

プログラムデザイン時に、プロパティウィンドウで設定してもかまいませんし、プログラム実行時であれば、フォームのLoadイベントプロシージャで設定してもかまいません。設定する値は、ミリ秒（ms）単位の時間になります。

タイマーが動作したときの処理は、タイマーコントロールのTimerイベントプロシージャに記述します。

4　では、順を追ってコードを説明していきましょう。まず、Form1のコードウィンドウを開きます。宣言セクション「General」「Declarations」に、モジュールレベル変数「ImgNo」を宣言します。この変数は、イメージコントロールに表示するビットマップファイル名を作成するのに使用します。

```
Dim ImgNo As Integer
```

5　次に、フォームの初期化プロシージャ「Load」に、この変数の初期値を設定します。ビットマップファイル名は「anime1.bmp」と1から始まっていますので、「1」を代入します。

```
ImgNo = 1
```

そして、タイマーコントロールの［Interval］プロパティを「100」（100ms）に設定します。これで、タイマーコントロールのTimerイベントは、0.1秒単位で発生します。

```
Timer1.Interval = 100
```

また、フォームに配置した「StartBtm」ボタンを無効にする処置をしておきます。このボタンは、アニメーションが停止したときのみ有効になるボタンにしておきます。

```
StartBtn.Enabled = False
```

このように、状況に応じてボタンの有効・無効を切り替えて使うには、［Enabled］プロパティを使用します。

```
Private Sub Form_Load()
    ImgNo = 1
    Timer1.Interval = 100
    StartBtn.Enabled = False
End Sub
```

6 タイマーコントロールの［Interval］プロパティに値がセットされると、タイマーコントロールは動作を始めます。そして、この［Interval］に設定された時間がくると（このプログラムでは0.1秒ごとに）、Timerイベントプロシージャが呼び出されます。

Timerイベントプロシージャでは、呼び出されるたびにビットマップファイルを、順番に読み込んで表示していきます。20枚のビットマップファイルが、0.1秒ごとに読み込まれるのです。これで、ユーザーの目にはイメージがアニメーションしているように見えるわけです。

7 まず、イメージコントロールにセットするビットマップファイルのファイル名を作成します。ビットマップファイルの名前は、すべて「animeXX.bmp」と、文字「anime」と「XX」という数字を組み合わせて作成してあります。そこで、文字「anime」と変数「ImgNo」、そして拡張子「.bmp」を使って、ファイル名を作成します。

ファイルを参照するためにはさらに「パス」が必要なのですが、ここでは完全にパス名を記述するかわりに、「相対参照」を使ってより扱いやすくしてみます（ファイルの参照については「第5日　ファイルを操作する」で詳しく解説します）。

プログラム実行中の現在のパス名は［App］オブジェクトの［Path］プロパティを使って取得します。この［Path］プロパティは、プログラムの実行中のみ値が取得可能です。

例えば「Program Files」フォルダの中に実行中のプロジェクトがあるとすると、これで取得されるパス名は、「C:¥Program Files」という文字列になります。このため実際にファイル名の参照に使用するときは、パス名とファイル名を、ディレクトリの区切り文字「¥」でつなぎます（「"¥" & "anime"」と書いてもかまいません）。

これで、現在作業中のプロジェクトファイルがあるフォルダの「animeXX」ビットマップファイルを参照することができるようになります。あとは、作成したファイル名を変数「BMP」に格納します。

```
Dim BMP As String

BMP = App.Path & "¥anime" & ImgNo & ".bmp"
```

現在のパス名（App.Path）
ディレクトリの区切り文字"¥"
ファイル名（anime" & ImgNo & ".bmp"）

8 ファイル名ができたら、LoadPicture関数でイメージコントロールの［Picture］プロパティに設定しますが、その前に繰り返しビットマップを表示するための処置をしておきます。

```
If ImgNo = 20 Then
    ImgNo = 0
End If
```

この処置は、ファイル名が「anime20.bmp」までできたら、「anime1.bmp」に戻す処置で、変数ImgNoの値が20になったら0を入れ直しています。後の処理で1を追加するのでここでは0を入力することに注意してください。これで、「anime1.bmp」から「anime20.bmp」までのイメージファイルを繰り返し再生することができます。

9

実際にイメージコントロールにビットマップを表示するには、イメージコントロールの［Picture］プロパティにファイル名を設定するのですが、それには直接ファイル名を代入するのではなく、LoadPicture関数を使って代入します。

```
Image1.Picture = LoadPicture(BMP)
ImgNo = ImgNo + 1
```

これで0.1秒ごとにビットマップが1枚ずつ読み込まれ、表示されていきます。

10

アニメーション処理ができたら、次にコマンドボタンの処理を作成します。

まず、［StopBtn］ボタンですが、このボタンはタイマーコントロールを一旦停止させます。タイマーコントロールの動作を決めるのは、［Enabled］プロパティです。「True」で動作、「False」で停止します。

また、タイマーがストップしたら、StartBtnボタンを有効にし、再スタートに備えます。そして、すでにタイマーが停止していますから、StopBtnボタン自身を無効にしておきます。

```
Private Sub StopBtn_Click()
    Timer1.Enabled = False
    StartBtn.Enabled = True
    StopBtn.Enabled = False
End Sub
```

StopBtnボタン

11

StartBtnボタンが押されたときは、今度は逆の処理をします。

タイマーコントロールの［Enabled］プロパティをTrueに設定し、Timerをオンにします。また、すでにStartBtnボタンが押されているので、StartBtnボタンを無効にしStopBtnボタンを有効にします。

```
Private Sub StartBtn_Click()
    Timer1.Enabled = True
    StartBtn.Enabled = False
    StopBtn.Enabled = True
End Sub
```

StartBtn ボタン

12 ExitBtnボタンは、すでにおなじみになったEndステートメントを記述し、プログラムを終了させます。

```
Private Sub ExitBtn_Click()
    End
End Sub
```

ExitBtn ボタン

13 ［ファイル］メニューにある［終了］メニューのプロシージャを作成します。これも、Endステートメントを記述するだけです。

アニメーション
ファイル(F) 速度
終了(X)

```
Private Sub IDMExit_Click()
    End
End Sub
```

14 ［速度］メニューにある［設定］メニューのプロシージャでは、表示速度設定用のフォーム「Form2」を、Showメソッドを使って呼び出します。このとき、Form2が閉じないとForm1の操作ができないように、「モーダル」で表示します。

```
Private Sub IDMSet_Click()
    Form2.Show 1
End Sub
```

15 今度は、表示速度設定用のフォーム「Form2」のプロシージャを作成します。
このフォームでは、Form1に配置したタイマーコントロールのIntervalプロパティを、Form2から操作します。
実際には、［閉じる］ボタンを押したときに、［Interval］プロパティに値を設定するようにしますが、速度をいくつにするのかは、オプションボタンコントロールで行います。ここではオプションボタンから渡されてくる変数「Speed」を使って、プロパティの設定を作成しておきます。

また、Intervalプロパティを設定したら、Form2は必要なくなりますので、Unloadステートメントで閉じます。

```
Private Sub Command1_Click()
    Form1.Timer1.Interval = Speed
    Unload Form2
End Sub
```

●Form2からForm1のコントロールを操作するとき

```
Form1.Timer1.Interval = Speed
```

Form名を追加する／コントロールのプロパティ名

16 変数「Speed」は、オプションボタンコントロールのイベントプロシージャでも値の代入をするので、モジュールレベルの変数として、Form2の宣言セクションに記述しておきます。

```
Dim Speed As Integer
```

Project1 - Form2 (コード)

(General)　(Declarations)

```
Dim Speed As Integer
```

17 オプションボタンコントロールは、複数を配置すると、その中で必ず1つしか選択できないコントロールです。そして、ユーザーがオプションボタンを選ぶと、Clickイベントが発生しますので、その処理は、Clickイベントプロシージャに記述します。

このプログラムでは、タイマーコントロールの速度を選んでもらうので、プロシージャでは変数「Speed」にそのミリ秒を代入する式を記述します。ユーザーが、その速度のオプションボタンを選べば、変数Speedに値が代入され、［閉じる］ボタンを押した時点で、Form1のタイマーコントロールの［Interval］プロパティに代入されます。

```
Private Sub Option100_Click()
    Speed = 100
End Sub

Private Sub Option300_Click()
    Speed = 300
End Sub
```

```
Private Sub Option50_Click()
    Speed = 50
End Sub
```

18 最後に、フォームのLoadイベントプロシージャで、[Interval] プロパティに「100」を設定しておきます。これは、Form2の起動時に初期値を設定しておくためです。

19 以上で、プログラムは完成です。実際に動作させてください。アニメーションの速度は、数字が小さければ速く、数字が大きければ遅く動作します。

まとめ

タイマーコントロールは、アニメーション以外にもいろいろな使い道が考えられます。使う上でのポイントは、必ず [Interval] プロパティに速度を設定することと、Timerイベントプロシージャで [Interval] プロパティを設定するのではなく、別のプロシージャで設定することです。

また、オプションボタンコントロールは、Windowsのアプリケーションでもおなじみのコントロールです。複数の中から1つだけ選択させたい場合に使うと有効なコントロールです。

練習問題

Q1 ラベルコントロールを使って、文字がフォームの右から左に流れていくプログラムを作成してください。

Form1

Welcome!

【ヒント】

①フォームに、Timerコントロールとラベルコントロールを配置します。
②ラベルコントロールの表示位置は、上下方向が [Top] プロパティ、左右方向が [Left] プロパティになります。
③Timerイベントプロシージャで、ラベルコントロールの表示位置を変えれば、文字が移動しているように見えます。

解答は巻末に→

ワンポイントアドバイス

メニューコントロールを作成するときに、そのメニューを選ぶとダイアログボックスが表示されるメニューの場合は、キャプションの文字の末尾に「...」を付けるのが慣例になっています。

これは、ユーザーがメニューを選んだときに、「このメニューを選ぶとダイアログボックスが表示されます」ということを伝えるための、目印になっています。

ファイル(F)　編集(E)　表示(V)　プロジェクト(P)　書式(O)
新しいプロジェクト(N)　Ctrl+N
プロジェクトを開く(O)...　Ctrl+O

メニュー名の後ろに「...」が付いている

COLUMN

グラフィック作成ソフト

Visual Basicでグラフィック（イメージ）ファイルを作成する場合は、専用のグラフィックソフトを使うと綺麗なグラフィックが作成できます。

Windows95/98には「ペイント」というソフトが付属しています。このソフトを使えば、簡単なイラストを作成できます。ただし、色数に制限があるのと、あまり凝ったデザインはできません。Adobe社の「イラストレーター」というソフトは、プロも使うイラスト作成ソフトで、高度な編集操作と豊富な特殊効果機能を持っています。ただし、その分価格が高めになっています。3DCG作成ソフトを使うと、今回の例のようなリアルなイメージを作成できます。また、アニメーション作成機能を持っていますので、作成したCGオブジェクトを簡単にアニメーションにできます。作成したアニメーションは、連続写真のように1枚ずつビットマップファイルに変換できますから、それをVisual Basicで使えるようになります。

筆者は、ソニーテクトロニクス社の「STmage3Dデザイン」というソフトでこのレッスンの画像を作成しました。最近は、たくさんの3DCGソフトが販売されていますから、アニメーション機能を持ったものを購入するといいでしょう。

Adobeイラストレーター

STmage3Dデザインでのアニメーション作成

Lesson 19

第４日
２時限目

コントロールの連携と動作

Lesson19.vbp
VB6CD
Lesson19

◎Learning Edition

スクロールバーとツールチップを使う

スクロールバーとテキストボックスコントロールの使い方を理解します。また、ツールチップの設定方法を覚えます。

今回作成するプログラム

アニメーション
ファイル(F)　速度
プログラムの終了
アニメーションプログラム
表示速度
100

1 ボタンにマウスポインタを置くと…
2 ツールチップが表示される
3 スクロールバーをクリックすると…

アニメーション
ファイル(F)　速度
アニメーションプログラム
表示速度
500

4 テキストボックスの値が変わる

POINT

前回作成した「Lesson18.vbp」に、スクロールバーとテキストボックスコントロールを追加し、これらのコントロールでタイマーの速度を設定できるようにします。

また、コマンドボタンなどに「ツールチップ」と呼ばれる小さな説明を表示する機能を追加します。

コントロールとプロパティの設定

1 「Lesson18.vbp」を開く

2 ツールボックスの［Label］ボタン A をクリックし、フォーム上にラベルコントロールを配置する

アニメーション
ファイル(F)　速度
アニメーションプログラム
Label2

1 ここをクリックして…

2 そのままここまでドラッグ

3 プロパティウィンドウで［Caption］プロパティの値を「表示速度」に変更する

プロパティ - Label2
Label2 Label
全体 | 項目別

BackColor	&H8000000F&
BackStyle	1 - 不透明
BorderStyle	0 - なし
Caption	表示速度
DataField	
DataFormat	
DataMember	
DataSource	

1 ここをダブルクリック

2 「表示速度」を入力

Lesson18.vbp の保存場所

このファイルは、付録CD-ROMの「Lesson18」フォルダに収録されています。

ラベルを選択

この操作を行う前に、作成したラベルコントロールが選択されていることを確認してください。

値の切り替え

[BorderStyle] プロパティをダブルクリックして、[0 - なし] と [1 - 実線] を切り替えることもできます。

テキストボックスを選択

この操作を行う前に、作成したテキストボックスコントロールが選択されていることを確認してください。

4 [BorderStyle] プロパティの値を [1 - 実線] に設定する

1 ここをクリック
2 ここをクリック
3 ここをクリック

5 ツールボックスの [TextBox] をクリックし、フォーム上にテキストボックスコントロールを配置する

1 ここをクリックして…
2 そのままここまでドラッグ

6 プロパティウィンドウで [Text] プロパティの値を「100」に設定する

1 ここをダブルクリック
2 「100」を入力

7 [ToolTipText] プロパティに「表示速度の設定」を入力する

プロパティ - Text1
Text1 TextBox
全体 | 項目別

TabStop	True
Tag	
Text	100
ToolTipText	表示速度の設定
Top	2310
Visible	True
WhatsThisHelpID	0
Width	600

1 ここをダブルクリック
2 値を入力

8 [MaxLength] プロパティの値を「3」に設定する

プロパティ - Text1
Text1 TextBox
全体 | 項目別

Locked	False
MaxLength	3
MouseIcon	(なし)
MousePointer	0 - 既定値
MultiLine	False
OLEDragMode	0 - 手動
OLEDropMode	0 - なし
PasswordChar	

1 ここをダブルクリック
2 「3」を入力

9 ツールボックスで [HScrollBar] をクリックし、フォーム上にスクロールバーコントロールを配置する

アニメーション
ファイル(F)　速度
アニメーションプログラム
表示速度
100

1 ここをクリックして…
2 そのままここまでドラッグ

MaxLength

値の最大値です。テキストボックスの [MaxLength] を「3」に設定することで、3桁以上の値の入力を受け付けなくします。

スクロールバーを選択

この操作を行う前に、作成したスクロールバーコントロールが選択されていることを確認してください。

10 プロパティウィンドウでプロパティ［Max］の値を「999」、［Min］の値を「1」に設定する

プロパティ - HScroll1
HScroll1 HScrollBar
全体 項目別

LargeChange	100
Left	3315
Max	999
Min	1
MouseIcon	(なし)
MousePointer	0 - 既定値
RightToLeft	False
SmallChange	10

1 「999」を入力
2 「1」を入力

11 ［LargeChange］プロパティの値を「100」に設定する

プロパティ - HScroll1
HScroll1 HScrollBar
全体 項目別

HelpContextID	0
Index	
LargeChange	100
Left	3315
Max	1000
Min	1

1 ここをダブルクリック
2 「100」を入力

12 ［Value］プロパティの値を「100」に設定する

プロパティ - HScroll1
HScroll1 HScrollBar
全体 項目別

TabStop	True
Tag	
Top	2295
Value	100
Visible	True
WhatsThisHelpID	0
Width	1710

1 ここをダブルクリック
2 「100」を入力

13 [Stop] ボタンをクリックして選択する

1 このボタンをクリック

14 [ToolTipText] プロパティに「アニメーションの停止」を入力する

1 ここをダブルクリック

2 値を入力

15 同様に、[Start] ボタンと [終了] ボタンの [ToolTipText] プロパティに次の値を設定する

ボタン名	[ToolTipText] の値
[StartBtn]	アニメーションの開始
[ExitBtn]	プログラムの終了

[StartBtn] ボタンと
[ExitBtn] ボタン

16 完成したレイアウト

コードの記述

※「Lesson18.vbp」のコードに追加します。

Form1:HScroll1-Change

Project1 - Form1 (ｺｰﾄﾞ)

HScroll1 / Change

```
Private Sub HScroll1_Change()
    Text1.Text = HScroll1.Value
    Timer1.Interval = HScroll1.Value
End Sub
```

Form1:Text1-Change

Project1 - Form1 (ｺｰﾄﾞ)

Text1 / Change

```
Private Sub Text1_Change()
    If Text1.Text = "" Or Text1.Text = "0" Then
        MsgBox "数値を入力してください"
        HScroll1.Value = 1
        Text1.Text = "1"
    Else
        HScroll1.Value = Int(Text1.Text)
    End If
End Sub
```

Form2:Command1-Click

Project1 - Form2 (コード)

Command1　Click

```
Private Sub Command1_Click()
    Form1.HScroll1.Value = Speed
    Unload Form2
End Sub
```

解　説

1　このプログラムでは、アニメーションの表示速度の設定を、テキストボックスとスクロールバーコントロールで行えるようにしました。

これらのコントロールは、互いに連動して動作します。

①ユーザーが直接テキストボックスコントロールに速度を入力すると、Timerコントロールの［Interval］プロパティの値を変えます。

②ユーザーがスクロールバーを操作すると、テキストボックスコントロールの数字が変化し、Timerコントロールの［Interval］プロパティの値が変わります。

③すでに作成している、表示速度設定用のオプションボタンを使って表示速度を設定すると、テキストボックスとスクロールバーコントロールもその値に変化します。

2　テキストボックスコントロールは、ユーザーから文字の入力を受け付けるコントロールです。入力フィールドに入力された文字を、［Text］プロパティに格納します。通常は、一行入力に設定されており、［MaxLength］プロパティに設定された文字数を受け付けます。

このプログラムでは、3桁までの数字を受け付けるようにしていますから、［MaxLength］プロパティを「3」に設定しました。

3　テキストボックスコントロールの入力フィールドに、ユーザーから文字が入力されると、Changeイベントが発生します。

このプログラムでは、入力があるとすぐに水平スクロールバーとTimerコントロールに、その数字を渡すように作成してありますから、Changeイベントプロシージャでこれらの処理を行っています。

```
Private Sub Text1_Change()

End Sub
```

4 テキストボックス（Text1）にユーザーから入力があると、その値は［Text］プロパティに格納されますので、そのままスクロールバーコントロール（HScroll1）の［Value］プロパティに代入します。このとき、Int関数により文字列が数値（整数）データに変換されて入力されます。Int関数は指定した数値の整数部分を返す関数です。

```
HScroll1.Value = Int(Text1.Text)
```

5 しかし、テキストボックスになにも入力されないか、または「0」が入力されると、スクロールバーコントロールの［Value］プロパティは、空白または「0」が設定されてしまいます。この［Value］プロパティの値は、あとでTimerコントロールの［Interval］プロパティに設定しますので、「0」のままだとTimerコントロールは動作できなくなって、エラーを起こしてしまいます。それを防止するために、事前に［Text］プロパティの値をチェックして、Ifステートメントで注意メッセージを表示する処理を行っています。

```
Private Sub Text1_Change()
    If Text1.Text = "" Or Text1.Text = "0" Then
        MsgBox "数値を入力してください"
        HScroll1.Value = 1
        Text1.Text = "1"
    Else
        HScroll1.Value = Int(Text1.Text)
    End If
End Sub
```

「Or」は論理演算子で、演算子の前後の式のどちらかが真であれば、真を返します。ここでは、［Text］プロパティの値が「""」（なし）か「0」の場合だけ、Ifステートメントの式を有効にしています。

```
If Text1.Text = "" Or Text1.Text = "0" Then
```

6 水平スクロールバーコントロールは、ユーザーによって矢印またはバー部分がクリックされると、スライダーを動かすと同時に［Value］プロパティにその移動量を数値データで格納します。

スクロールバーを設定する際は、最小・最大の移動量などのプロパティを設定します。

プロパティ	説明
[Max]	スクロール範囲の上限値（最大値）を設定します。
[Min]	スクロール範囲の下限値（最小値）を設定します。 スクロールバーは、このMaxとMinの間を移動します。
[SmallChange]	スクロールバーの矢印ボタンを押したときに、スライダーが移動する移動量を設定します。
[LargeChange]	スクロールバーのバーの部分を押したときに、スライダーが移動する移動量を設定します。

[Min]
[Max]
[LargeChange]
[SmallChange]

アニメーション
ファイル(F)　速度
アニメーションプログラム
表示速度
130

ここをクリックするとSmallChangeで設定された量だけ動く

アニメーション
ファイル(F)　速度
アニメーションプログラム
表示速度
330

ここをクリックするとLargeChangeで設定された量だけ動く

7 スクロールバーの矢印ボタンやバーの部分が押されたり、スライダーが動かされると、Changeイベントが発生し、移動量は［Value］プロパティに格納されます。

そこで、ユーザーによってスクロールバーがクリックされると、Changeイベントプロシージャで、HScrollコントロールの［Value］プロパティの値を、Timerコントロールの［Interval］プロパティに代入しています。

その際、この値がユーザーにもわかるように、テキストボックスコントロールの［Text］プロパティに代入して表示しています。

```
Private Sub HScroll1_Change()
    Text1.Text = HScroll1.Value
    Timer1.Interval = HScroll1.Value
End Sub
```

これで、スクロールバーがクリックされると、アニメーションの表示速度が変化し、テキストボックスにも表示されるようになります。

8 Form2では、表示速度をオプションボタンで設定していましたが、これもユーザーが選んだ値をスクロールバーに反映させておきます。これは、変数Speedをそのままスクロールバーの［Value］プロパティに代入するだけです。これで、テキストボックスコントロールの数字も変更されます。

```
Private Sub Command1_Click()
    Form1.HScroll1.Value = Speed
    Unload Form2
End Sub
```

追加した処理（Form1.HScroll1.Value = Speed）

9 なぜ、スクロールバーの［Value］プロパティに値を代入しただけで、テキストボックスも表示が変化するのでしょう。また、Timerコントロールの［Interval］プロパティへの値の代入は、スクロールバーコントロールのChangeイベントプロシージャにしか記述していないのに、なぜテキストボックスコントロールに入力しただけで、表示速度が変化するのでしょう

実は、スクロールバーコントロールの（HScroll1）［Value］プロパティに値が代入されると、Changeイベントが発生するのです。ユーザーがスクロールバーを操作しても、コードで値を代入してもChangeイベントが発生します。

スクロールバーにChangeイベントが発生すると、Changeイベントプロシージャが呼び出され、そこに記述されている処理を実行します。動作を整理すると、次のようになります。

スクロールバーの処理と Change イベントの関係

①ユーザーがテキストボックスコントロールを操作した場合

スクロールバーコントロールの Value プロパティに値が代入される

スクロールバーコントロールの Change イベントが発生

Interval プロパティへの値の代入が実行される

②ユーザーがスクロールバーを操作した場合

テキストボックスコントロールの Text プロパティに値が代入

スクロールバーコントロールの Change イベントが発生

Interval プロパティへの値の代入が実行される

③ Form2 のオプションボタンが選択され「閉じる」ボタンが押された場合

スクロールバーコントロールの Value プロパティに値が代入される

スクロールバーコントロールの Change イベントが発生

テキストボックスコントロールの Text プロパティに値が代入

Interval プロパティへの値の代入が実行される

10

ユーザーがボタンなどの上にマウスを置くと、小さな説明文が表示されます。Windows 95/98ではおなじみの機能ですが、これを「ツールチップ」といいます。

この機能は、コントロールにある［ToolTipText］プロパティに表示したい文字を設定すると実行されます。すべてのコントロールが［ToolTipText］プロパティを持っているわけではありませんから注意してください。

ツールチップの表示

11 以上でプログラムが完成です。実行させて、設定したテキストボックスやスクロールバーで、アニメーションの表示速度を変えてみてください。

COLUMN

フォームの有効期間

フォームは、いつ作成表示され、いつ終了するのでしょうか。一般的なプログラムを実行すると、フォームは①作成②ロード③表示④解放（アンロード）、の4つの状態を通常移行していきます。目に見える部分ではフォームが表示された③のときに作成されたように思えますが、それより前の①〜②の短い段階で既にフォームは存在しています。

イベントレベルでは、まず①のときはInitializeイベントが発生し、②でLoadイベント、③でウィンドウが最前面に表示されればActivateイベント、④になったときQueryUnloadイベント→Unloadイベント→ Terminateイベントという順序で発生した後、終了し完全に解放されます。これらをわかりやすく見せるためのサンプルプログラム「Column19.vbp」を「Lesson19」フォルダに収録しているので、起動して確かめてみてください。

フォームを任意の時期に表示する場合にはShowメソッドを使います。非表示にするにはHideメソッドです。ただし、非表示にしてもメモリにフォームの情報は残っています。メモリからすべてを削除するには、Unloadステートメントを使用してください。

Endステートメントはどちらかというと強制終了的な命令です。たとえば複数のフォームがあるようなプログラムでEndステートメントを実行するとプログラム全体が終了してしまいます。こういった場合は、フォームをいったんUnloadステートメントでメモリから削除してから、プログラムを終了するようにします。

また、今までのプログラムではユーザーがウィンドウの右上の［閉じる］ボタンをクリックすると、強制的に終了していましたが、サンプルプログラム「Column19.vbp」ではUnload（またはQueryUnload）イベントを利用して、ユーザーに終了させるかどうかを確認する簡単なコードサンプルも掲載しています。自分のプログラムに応用するとよいでしょう。

まとめ

表示速度の設定１つをとっても、

①メニューで操作する
②ポップアップメニューを使う
③ダイアログボックスで操作する
④オプションボタンを使う
⑤テキストボックスを使う
⑥スクロールバーを使う

など、いろいろなユーザーインターフェイスを作成することができます。状況に応じて一番ユーザーが使いやすいと思われるコントロールやインターフェイスを作成することが大事ですね。
　また、コントロールの連携は、プロパティとイベントを組み合わせるのが、ポイントです。

練習問題

Q1　「Q&A-18.vbp」のプログラムに、Lesson19.vbpと同様にテキストボックスとスクロールバーコントロールを付けて、文字の表示速度を変えられるようにしてください。

Form1
Welcome!
150

解答は巻末に→

ワンポイントアドバイス

ツールチップに使う文字は、なるべく簡潔にします。また、的確に操作内容がわかるような表現を使うようにしましょう。

Lesson 20

第４日
３時限目

コントロールの連携と動作

Lesson20.vbp
VB6CD
Lesson20

◎Learning Edition

チェックボックス・リストボックスコントロールを使う

チェックボックス・リストボックスコントロールの使い方を覚えます。また、これらを組み合わせて使うテクニックを学びます。

今回作成するプログラム

リスト選択プログラム
名前の選択
リストから選択
名前を入力して下さい。
表示
麻美
明日香
久美子
絵美
友子
綾香

1 チェックを付け…
2 リストから名前を選択し…
3 このボタンをクリックすると…

リスト選択プログラム
名前の選択
リストから選択
名前を入力して下さい。
表示
麻美
明日香
久美子
絵美
友子
綾香
麻美が選ばれました。

4 メッセージが表示される

POINT

　チェックボックスコントロールとリストボックスコントロールを組み合わせて、簡単な選択ダイアログボックスを作成します。
　ユーザーからの入力を、テキストボックスとリストボックスの２とおり用意し、チェックボックスで切り替えて使います。
　ユーザーに、複数の項目から１つだけ選択してもらう場合は、リストボックスコントロールを使うと便利です。

コントロールとプロパティの設定

※新しいプロジェクトとフォームを用意します。

1 ツールボックスの［Label］をクリックし、フォーム上にラベルコントロールを配置する

2 プロパティウィンドウで［Caption］プロパティの値を「名前の選択」に変更する

3 ツールボックスの［TextBox］をクリックし、フォーム上にテキストボックスを配置する

ラベルを選択

この操作を行う前に、作成したラベルコントロールが選択されていることを確認してください。

テキストボックスを選択

この操作を行う前に、作成したテキストボックスコントロールが選択されていることを確認してください。

4 プロパティウィンドウで［Text］プロパティの値を「名前を入力してください。」に変更する

5 ツールボックスの［CheckBox］をクリックし、フォーム上にチェックボックスを配置する

チェックボックスを選択

この操作を行う前に、作成したチェックボックスコントロールが選択されていることを確認してください。

6 プロパティウィンドウで［Caption］プロパティの値を「リストから選択」に変更する

7 ツールボックスの［ListBox］をクリックし、フォーム上にリストボックスを配置する

8 プロパティウィンドウで［List］プロパティの値に10人分の名前を入力する

コンボボックスと注意する

［ListBox］ボタンは［ComboBox］ボタンと形が似ているので注意してください。

［ListBox］

［ComboBox］

リストボックスを選択

この操作を行う前に、作成したリストボックスコントロールが選択されていることを確認してください。

リストボックスのスクロールバー

リストボックスコントロールに表示しきれない行数の値を入力すると、リストボックスに自動的に垂直のスクロールバーが追加されます。

9 プロパティウィンドウで［Enabled］プロパティの値を「False」に設定する

DragIcon	(なし)
DragMode	0 - 手動
Enabled	False
Font	True
ForeColor	False
Height	1320

1 ここをクリック
2 ここをクリック
3 ここをクリック

10 ツールボックスの［CommandButton］ を クリックし、コマンドボタンを配置する

1 ここをクリックして…
2 そのままここまでドラッグ

11 プロパティウィンドウで［Caption］プロパティの値を「表示」に変更する

(オブジェクト名)	Command1
Appearance	1 - 3D
BackColor	&H8000000F&
Cancel	False
Caption	表示
CausesValidation	True
Default	False

1 ここをダブルクリック
2 「表示」を入力

値の切り替え

［Enabled］プロパティをダブルクリックして、［True］と［False］を切り替えることもできます。

コマンドボタンを選択

この操作を行う前に、作成したコマンドボタンコントロールが選択されていることを確認してください。

12 ツールボックスの［Label］ A をクリックし、フォーム上にラベルコントロールを配置する

Form1
名前の選択
リストから選択
名前を入力してください。
麻美
明日香
久美子
絵美
友子
綾香
表示
Label2

1 ここをクリックして…
2 そのままここまでドラッグ

13 プロパティウィンドウで［Caption］プロパティの値を削除する

プロパティ - Label2
Label2 Label
全体 | 項目別

BackColor	&H8000000F&
BackStyle	1 - 不透明
BorderStyle	0 - なし
Caption	
DataField	
DataFormat	

1 ここをダブルクリック
2 Del キーを押す

ラベルを選択

この操作を行う前に、作成したラベルコントロールが選択されていることを確認してください。

14 ［BorderStyle］プロパティの値を［1 - 実線］に設定する

プロパティ - Label2
Label2 Label
全体 | 項目別

BackColor	&H8000000F&
BackStyle	1 - 不透明
BorderStyle	1 - 実線
Caption	
DataField	
DataFormat	

1 セルをダブルクリック
2 値が切り替わる

値の切り替え

セルの右端にある［▼］ボタンをクリックし、一覧から選択して値を切り替えることもできます。

フォームの選択

フォームを選択するには、フォーム上のコントロール以外の部分をクリックします。

15 Form1を選択し、[BorderStyle] プロパティの値を [1 - 固定（実線)] に変更する

Form1 Form

全体 | 項目別

Appearance	1 - 3D
AutoRedraw	False
BackColor	&H8000000F&
BorderStyle	1 - 固定(実線)
Caption	0 - なし
ClipControls	1 - 固定(実線)
ControlBox	2 - 可変
DrawMode	3 - 固定ﾀﾞｲｱﾛｸﾞ
DrawStyle	4 - 固定ﾂｰﾙ ｳｨﾝﾄﾞｳ
DrawWidth	5 - 可変ﾂｰﾙ ｳｨﾝﾄﾞｳ
	1

1 ここをクリック
2 ここをクリック
3 ここをクリック

16 [Caption] プロパティの値を「リスト選択プログラム」に変更する

Form1 Form

全体 | 項目別

(ｵﾌﾞｼﾞｪｸﾄ名)	Form1
Appearance	1 - 3D
AutoRedraw	False
BackColor	&H8000000F&
BorderStyle	1 - 固定(実線)
Caption	リスト選択プログラム
ClipControls	True

1 ここをダブルクリック
2 値を入力

17 完成したレイアウト

リスト選択プログラム

名前の選択

リストから選択

名前を入力して下さい。

麻美
明日香
久美子
絵美
友子
綾香

表示

コードの記述

Form1:(General)-(Declarations)

Project1 - Form1 (ｺｰﾄﾞ)
(General) (Declarations)

```
Dim CheckPos As Integer
```

Form1:Check1-Click

Project1 - Form1 (ｺｰﾄﾞ)
Check1 Click

```
Private Sub Check1_Click()

    CheckPos = Check1.Value
    Select Case CheckPos
        Case 1
            List1.Enabled = True
            Text1.Enabled = False
        Case 0
            List1.Enabled = False
            Text1.Enabled = True
    End Select
End Sub
```

Form1:Command1-Click

Project1 - Form1 (ｺｰﾄﾞ)
Command1 Click

```
Private Sub Command1_Click()
    Dim Msg As String

    Select Case CheckPos
        Case 1
            Msg = List1.Text & "が選ばれました"
        Case 0
            Msg = Text1.Text & "が選ばれました"
    End Select

    Label2.Caption = Msg
End Sub
```

解説

1 このプログラムは、ユーザーがテキストボックスに入力するか、リストボックスから選択した名前を、［表示］ボタンが押された時点で、ラベルコントロールに表示するプログラムです。

プログラム起動時は、テキストボックスで入力するようになっていますが、チェックボックスをオンにするとリストボックスから選択できるようになります。これは、Windowsのアプリケーションで表示されるダイアログボックスによく使われるテクニックです。

2 プログラムの処理を説明しておきます。

①プログラム起動時は、テキストボックスが有効でリストボックスは無効になっている
②ユーザーがテキストボックスに名前を入力して、「表示」ボタンを押すと、ラベルコントロールに名前が表示される
③「リストから選択」チェックボックスをオンにすると、テキストボックスは無効になり、リストボックスが有効になる
④リストボックスから名前を選択して［表示］ボタンを押すと、ラベルコントロールに名前が表示される

3 チェックボックスコントロールは、ユーザーがクリックするとチェックマークが付けられます。このとき、チェックボックスコントロールにはClickイベントが発生し、［Value］プロパティに「1」が設定されます。もう一度クリックすると、チェックがはずされ［Value］プロパティには「0」が設定されます。

リスト選択プログラム
名前の選択
リストから選択
名前を入力して下さい。
麻美
明日香
久美子
絵美
友子
綾香
表示
チェックが付いたチェックボックス

4　チェックボックスCheck1では、Clickイベントプロシージャで、現在チェックボックスにチェックが付いているかどうかを、［Value］プロパティの値を使って調べます。このとき、［Value］プロパティの値を、事前に変数CheckPosに格納しておきます。そして、チェックされていればテキストボックスを無効にして、リストボックスを有効にします。チェックされていなければ、テキストボックスを有効にして、リストボックスを無効にします。

有効・無効の切り替えは、それぞれのコントロールの［Enabled］プロパティを使います。

```
Private Sub Check1_Click()

    CheckPos = Check1.Value
    Select Case CheckPos
        Case 1
            List1.Enabled = True
            Text1.Enabled = False
        Case 0
            List1.Enabled = False
            Text1.Enabled = True
    End Select
End Sub
```

5　［Value］プロパティの値（チェックのオン・オフ）で処理を分岐するのに、Select Caseステートメントを使用しています。

Select Caseステートメントは、式（または変数）の値によっていくつでも処理を分岐できる条件分岐ステートメントです。

【Select Caseステートメントの書式】

```
    Select Case 式
          Case 式の値1
                処理1
          Case 式の値2
                処理2
          Case 式の値3
                処理3
          ....
          ....
    End Select
```

このプロシージャでは、変数CheckPosの値（すなわち［Value］プロパティの値）が1の場合と0の場合で2通りの処理を作成しています。

6 現在のチェックボックスのチェックの有無がわかれば、テキストボックスとリストボックスの、どちらから名前を取得すればいいのかがわかりますから、あとはその名前をラベルコントロールに出力するだけです。

これも同様に、Select Caseステートメントを使って、2つの処理を作成しています。つまり、チェックボックスがチェックされていれば、リストボックスから選択された名前を取得し、チェックされていなければテキストボックスから入力された名前を取得します。

```
Private Sub Command1_Click()
    Dim Msg As String

    Select Case CheckPos
        Case 1
            Msg = List1.Text & "が選ばれました。"
        Case 0
            Msg = Text1.Text & "が選ばれました。"
    End Select

    Label2.Caption = Msg
End Sub
```

7 リストボックスでユーザーがリストから選択した項目は、リストボックスコントロールの［Text］プロパティに格納されます。この値を調べることで、選択された項目名を把握できます。リストボックスに項目を設定する場合は、プログラムデザイン時は、プロパティリストにある［List］プロパティで設定します。このリスト項目は、プログラム実行時に追加と削除ができます（この処理は、次のLesson21で行います）。

また、テキストボックスに入力された文字も、テキストボックスコントロールの［Text］プロパティの値から取得できます。

8 以上で完成です。プログラムを実行してチェックボックスをクリックしてください。テキストボックスとリストボックスの有効・無効が交互に切り替わります。また、リストボックスで選択した名前が、ラベルコントロールに表示されます。

まとめ

チェックボックスコントロールは、オプションボタンコントロールと違い、単独でオン･オフが設定できるコントロールです。表示を切り替えたり、有効・無効を設定するなど、多くの場面で使われるコントロールです。使い方はこのように簡単なので、いろいろと試してください。

練習問題

Q1　「Lesson20.vbp」にチェックボックスコントロールを１つ追加し、ラベルコントロールの表示が斜体文字になるようにしてください。

【ヒント】

ラベルコントロールの文字を斜体にするには、ラベルコントロールの[FontItalic] プロパティを「True」に設定します。

解答は巻末に→

ワンポイントアドバイス

Select Case ステートメントは、他方向に処理を分岐する場合に威力を発揮する、条件分岐ステートメントです。Select Case ステートメントでは、「Case」に当てはまらないものを一括して処理できるように、「Case Else」節が用意されています。これは、If ステートメントの「Else」と同じような働きをします。

この例題プログラムでは、２方向しか分岐しませんでしたが、もっと多数の分岐処理を作成する場合は、If ステートメントよりも記述がすっきりできるので、Select Case ステートメントを使うほうがいいでしょう。

Lesson 21

第４日
４時限目

コントロールの連携と動作

Lesson21.vbp
VB6CD
Lesson21

◎Learning Edition

コンボボックスでのリスト項目の追加と削除

リストボックスにリスト項目を追加したり削除するテクニックと、メッセージの様々な表示方法を覚えます。

今回作成するプログラム

ガールフレンド名簿
真美
選択　追加　削除
Lesson21
リストボックスに追加しますか？
はい(Y)　いいえ(N)

1 彼女の名前を入力し…
2 ［追加］ボタンをクリックすると…
3 メッセージが表示される
4 ［はい］ボタンをクリックすると…

ガールフレンド名簿
真美
奈美子
由佳
香苗
真美

5 名前がリストに追加される

!!! POINT

リストボックスとテキストボックスコントロールを合体させたコントロールに、コンボボックスコントロールがあります。

このレッスンでは、コンボボックスコントロールを使って、リストボックスにリスト項目を追加・削除するプログラムを作成します。

コントロールとプロパティの設定

※新しいプロジェクトとフォームを用意します。

1 Form1の［Caption］プロパティを「ガールフレンド名簿」に変更する

Form1 Form	
BackColor	&H8000000F&
BorderStyle	2 - 可変
Caption	ガールフレンド名簿
ClipControls	True
ControlBox	True

1 ここをダブルクリック
2 値を入力

2 ツールボックスの［ComboBox］をクリックし、フォーム上にコンボボックスを配置する

ガールフレンド名簿
Combo1

1 ここをクリックして…
2 そのままここまでドラッグ

3 プロパティウィンドウで［Text］プロパティの値を「彼女の名前を入力して下さい。」に変更する

Combo1 ComboBox	
TabIndex	0
TabStop	True
Tag	
Text	彼女の名前を入力して下さい。

1 ここをダブルクリック
2 値を入力

4 ［List］プロパティに名前のリストを設定する

プロパティ－Combo1

Combo1 ComboBox	
Left	150
List	(ﾘｽﾄ)
Locked	奈美子
MouseIcon	

1 ここをクリック
2 ここをクリック
3 名前を入力

コンボボックスを選択

この操作を行う前に、作成したコンボボックスコントロールが選択されていることを確認してください。

5 ツールボックスの［CommandButton］をクリックし、フォーム上にコマンドボタンを3つ配置する

ガールフレンド名簿
彼女の名前を入力して下さい。
Command1 Command2 Command3

1 それぞれコピーして配置する

プロパティの設定

各ボタンを選択してから、［(オブジェクト名)］と［Caption］プロパティを設定してください。

6 各コマンドボタンの［(オブジェクト名)］プロパティと［Caption］プロパティを次の値に変更する

コマンドボタン	オブジェクト名	Caption
Command1	ChoiceBtn	選択
Command2	AddBtn	追加
Command3	DelBtn	削除

ガールフレンド名簿
彼女の名前を入力して下さい。
選択 追加 削除

ボタン名が設定された

7 ツールボックスの［Label］をクリックし、フォーム上にラベルコントロールを配置する

ガールフレンド名簿
彼女の名前を入力して下さい。
選択 追加 削除
Label1

1 ここをクリックして…

2 そのままここまでドラッグ

8 プロパティウィンドウで［BorderStyle］プロパティの値を［1 - 実線］に設定する

BackColor	&H8000000F&
BackStyle	1 - 不透明
BorderStyle	1 - 実線
Caption	Label1

1 セルをダブルクリック
2 値が切り替わる

値の切り替え

セルの右端にある［▼］ボタンをクリックし、一覧から選択して値を切り替えることもできます。

9 プロパティウィンドウで［Caption］プロパティの値を削除する

BackColor	&H8000000F&
BackStyle	1 - 不透明
BorderStyle	1 - 実線
Caption	

1 ここをダブルクリック
2 Delキーを押す

10 完成したレイアウト

ガールフレンド名簿

彼女の名前を入力して下さい。

選択　追加　削除

コードの記述

Form1:ChoiceBtn-Click

Project1 - Form1 (コード)

ChoiceBtn　Click

```
Private Sub ChoiceBtn_Click()
    Label1.Caption = Combo1.Text & "が選択されました。"
```

Form1:AddBtn-Click

Project1 - Form1 (ｺｰﾄﾞ)

AddBtn | Click

```
Private Sub AddBtn_Click()
    Dim Result As Integer
    Dim Name As String

    Name = Combo1.Text
    Result = MsgBox("リストボックスに追加しますか？", vbYesNo)
    If Result = vbYes Then
        Combo1.AddItem Name
    End If
End Sub
```

Form1:DelBtn-Click

Project1 - Form1 (ｺｰﾄﾞ)

DelBtn | Click

```
Private Sub DelBtn_Click()
    Dim Result As Integer
    Dim Name As String
    Dim Msg As String

    Name = Combo1.Text
    If Combo1.ListCount = 0 Then
        MsgBox "これ以上削除できません"
    Else
        Msg = "リストボックスから" & Name & " を削除しますか？"
        Result = MsgBox(Msg, vbYesNo)
        If Result = vbYes Then
            Combo1.RemoveItem Combo1.ListIndex
        End If
    End If
End Sub
```

解 説

1　コンボボックスコントロールは、テキストボックスコントロールとリストボックスコントロールが一体になったコントロールです。ユーザーからの文字入力を受け付けることができ、なおかつリストをドロップダウン表示させることができます。

例題プログラムは、コマンドボタンとコンボボックスを使って、ユーザーの選択したリスト項目をラベルコントロールに表示するとともに、ユーザーが入力した名前をリストに追加したり、リストから削除する操作をします。

彼女の名前を入力して下さい。
奈美子
由佳
香苗

コンボボックス

2　コンボボックスでユーザーが入力した文字、またはリストから選択した項目は、コンボボックスコントロールの［Text］プロパティに格納されます。このプロパティの値を調べることによって、ユーザーの選択を把握できます。

処理は、配置した3つのコマンドボタンの、Clickイベントプロシージャですべて行います。まず、［選択］ボタンでは、コンボボックスの［Text］プロパティの値を、そのままラベルコントロールの［Caption］プロパティに代入します。

```
Private Sub ChoiceBtn_Click()
    Label1.Caption = Combo1.Text & " が選択されました。"
End Sub
```

3　コンボボックスのリストに項目を追加するには、AddItemメソッドを使用します。

【AddItemメソッドの書式】

```
object.AddItem item, index
```

object	追加の対象となるオブジェクト名を記述します。
item	追加する文字列（項目名）を指定します。
index	追加する項目を配置する場所を番号で指定します。省略可。

オブジェクトにコンボボックスコントロールを指定し、AddItemのあとに追加する項目名を記述します。追加した項目名は、リストの最後尾に順次付け加えられていきます。リストの任意の位置に追加したい場合は、Indexに追加したい位置を番号で指定します。たとえば、リストの先頭に追加したければ、Indexを0に指定します。

また、コンボボックスコントロールの［Sorted］プロパティを「True」に設定すると、リスト項目は自動的に昇順で並べ替えられます。ただし、文字コード順に並ぶので、漢字は読みがなどおりに正しく並ばないことがあります。

4 このプログラムでは、［追加］ボタンのClickイベントプロシージャで項目の追加を行います。まず、コンボボックス（Combo1）に入力された内容を、その［Text］プロパティから取得し、変数Nameに代入しておきます。

そして、メッセージボックスを表示して、［はい］ボタンが押されたとき（Result=vbYesのとき）のみ、リストの追加処理を行うようにします。［いいえ］ボタンが押されたときは、何も処理をしません。

```
Name = Combo1.Text

Result = MsgBox("リストボックスに追加しますか？", vbYesNo)
If Result = vbYes Then
    Combo1.AddItem Name
End If
```

Lesson21

リストボックスに追加しますか？

はい(Y)　いいえ(N)

メッセージボックスで確認する

5 メッセージボックスを表示して、追加の確認を行うには、MsgBox関数にボタンの数を引数で付け加えます。

これまで、メッセージボックスは［OK］ボタンが付いたタイプしか使いませんでした。［はい］［いいえ］［キャンセル］などのボタンが付いたメッセージボックスを使うには、そのボタンタイプを引数として同時に指定します（P.193の表参照）。

```
MsgBox("リストボックスに追加しますか？", vbYesNo)
```

引数にボタンタイプを指定すると、メッセージボックス関数は実行時に、ユーザーが押したボタンの種類を戻り値として返してきます。これを、変数で受け止めて、［はい］ボタン（vbYes）が押されたときだけリストに項目を追加するように、Ifステートメントで処理を組み立てます（P.194の表参照）。

```
Result = MsgBox("リストボックスに追加しますか？", vbYesNo)
If Result = vbYes Then
    Combo1.AddItem Name
End If
```

<MsgBox 関数で使える主なボタンタイプと指定値>

記述指定（文字定数）	数値指定	表示されるボタンの内容
vbOKOnly	0	[OK] ボタンのみ
vbOKCancel	1	[OK] と [Cancel] ボタン
vbAbortRetryIgnore	2	[中止] [再試行] [無視] ボタン
vbYesNoCancel	3	[はい] [いいえ] [キャンセル]
vbYesNo	4	[はい] [いいえ]
vbRetryCancel	5	[再試行] [キャンセル]
vbCritical	16	[中止] アイコンと [OK] ボタン
vbQuestion	32	クエスチョンアイコンと [OK] ボタン
vbExclamation	48	Exclamation アイコンと [OK] ボタン
vbInformation	64	Information アイコンと [OK] ボタン

※メッセージボックスのタイプは、文字定数と数値のどちらを使ってもかまいません。
また、() 内で引数を指定した場合は、必ず変数で戻り値を受け取らないと、実行エラーで正常に動作しませんので注意してください。

●ボタンの表示例

< vbAbortRetryIgnore >

[中止] ボタン、[再試行] ボタン、
[無視] ボタン

< vbYesNo >

[はい] ボタン、[いいえ] ボタン

< vbQuestion + vbYesNoCancel >

アイコン付きメッセージボックス ([はい] ボタン、
[いいえ] ボタン、[キャンセル] ボタン)

<押されたボタンの戻り値>

記述指定（文字定数）	数値指定	押されたボタンの内容
vbOK	1	[OK] ボタン
vbCancel	2	[Cancel] ボタン
vbAbort	3	[中止] ボタン
vbRetry	4	[再試行] ボタン
vbIgnore	5	[無視] ボタン
vbYes	6	[はい] ボタン
vbNo	7	[いいえ] ボタン

6

今度は、リストから項目を削除する処理を組み立てます。

コンボボックス（またはリストボックス）で、リスト項目を参照するプロパティに［ListCount］と［ListIndex］プロパティがあります。［ListCount］プロパティは、そのリストボックスにいくつのリスト項目があるのか、総項目数を格納しているプロパティです。また、［ListIndex］プロパティは、現在選択されている項目が、リストの先頭から何番目なのかを格納しているプロパティです。これらのプロパティを使って、リストボックスの現在の状態を把握します。

まず最初に、リストに項目があるのかないのかを調べます。Combo1の［ListCount］プロパティの値が「0」であれば、リストは空ですから項目の削除はできません。そのため、このプロパティの値を調べ、リストが空であれば削除できないというメッセージを表示させます。

```
Name = Combo1.Text
If Combo1.ListCount = 0 Then
    MsgBox "これ以上削除できません"
```

7

［ListCount］プロパティが「0」でなければ、リストを削除できますので、その処理を組み立てます。

リストから項目を削除するには、RemoveItemメソッドを使います。このメソッドは、削除する項目を、リストの番号で指定します。追加に使ったAddItemは項目名で指定しましたが、削除の場合はリストの順番で指定します。そこで、現在選択されている項目の番号を［ListIndex］プロパティから取得します。そして、この番号を指定してRemoveItemメソッドを実行します。

このとき、追加の場合と同様に、メッセージボックスで削除を確認し、［はい］ボタンが押されたときだけ削除処理を実行します。

【RemoveItem メソッドの書式】

```
object.RemoveItem index
```

object	削除の対象となるオブジェクト名を記述します。
index	削除する項目を配置する場所を番号で指定します。

```
Else
    Msg = "リストボックスから" & Name & " を削除しますか"
    Result = MsgBox(Msg, vbYesNo)
    If Result = vbYes Then
        Combo1.RemoveItem Combo1.ListIndex
    End If
End If
```

8 以上で完成です。［追加］［削除］ボタンを使ってリスト項目を操作してください。現在リストにある項目を追加・削除することができますし、新しく入力した項目も追加・削除することができます。

COLUMN ……

処理を追加して動作を安全にする

このプログラムはエラー処理を考慮していません。例えば、起動した直後に［削除］ボタンをクリックし、メッセージボックスで［はい］ボタンをクリックすると、「プロシージャの呼び出し、または引数が不正です。」というエラーが表示されプログラムが止まってしまいます。

これはCombo1の［ListIndex］に無効な値「-1」が格納されているからです。空のリストボックスに対して［削除］ボタンをクリックときも「-1」が［ListIndex］プロパティに入力されます。このためRemoveItemメソッドが実行できないのでエラーになります。

このエラーを回避するには［はい］ボタンをクリックした後、さらに［ListIndex］プロパティの値を調べて処理を分岐します。例えば、

```
If Combo1.ListIndex < 0 then
    処理1
Else
    RemoveItemメソッド
End if
```

などの文を追加し、処理1で注意メッセージを表示すれば、RemoveItemの実行エラーを防ぐことができます（「Lesson21.vbp」のコメント部分を参照してください）。これはプログラムを正確に動作させるためのエラー処理の一例です。エラーの対処法には他にもいろいろとあります。エラー処理の実際についてはLesson32以降を参照してください。

なおこのプログラムでは、これらの操作はあくまでもプログラム実行時にその場限りで行われる操作であり、プログラムを終了するとリスト項目の変更は失われてしまいます。

リスト項目の変更を、次回プログラム起動時に復元するには、別のファイルに保存するなどの処理が必要になります（Lesson23を参照）。

まとめ

リストボックス・コンボボックスなどは、あらかじめ入力項目がわかっているもの（たとえば都道府県名とか取引先企業など）を選択するのに便利なコントロールです。ユーザーの入力操作を助け、操作性を向上させるのに使うと効果の高いコントロールですね。

また、コントロール間の連携も、操作するコントロールとプロパティを式で指定して行うので、さほど難しくありません。モジュールレベル変数と組み合わせて使うと、簡単にコントロール間の処理を組み立てることができます。

練習問題

Q1

「Lesson21.vbp」のプログラムを、最後のリスト項目が削除された時点で、［削除］ボタンを無効にするように変更してください。

【ヒント】

コマンドボタンの無効化は、［Enabled］プロパティを使います。これを、どこのプロシージャで処理するのかがポイントです。

また、［削除］ボタンを無効にしたら、今度項目が追加されたときに有効に戻さないと削除できません。この処理もどこで行うのか、考えてください。

解答は巻末に→

ワンポイントアドバイス

リストの追加・削除は、リストボックスコントロールもコンボボックスコントロールも、まったく同じメソッドを使って行えます。追加は「AddItem」、削除は「RemoveItem」です。また、リスト項目を参照するプロパティ「ListCount」、「ListIndex」も同様に使うことができます。

Let's VBSchool!!

第５日 (Lesson22～Lesson24)

ファイルを操作する

プログラムを作る上で欠かせないのが、ファイルを操作するテクニックです。設定した項目やデータなどをファイルに保存し、また読み込んで使用するためには、Visual Basicでファイルを操作する知識を身に付けておく必要があります。

しかし、だからといって難しいわけではありません。手軽さが売りもののVisual Basicらしく、とても簡単にファイルの読み書きが実行できるのです。

このレッスンでは、ファイルの読み書きと、リストボックス・ダイアログボックスからファイル名を取得する方法をマスターします。

今日のレッスンで解説する項目

- ❏ テキストファイル（文字だけのファイル）を読み込んで、他のコントロールに出力する操作手順
- ❏ ループ処理の作り方
- ❏ 独自のプロシージャを作成して、他のプロシージャから呼び出す方法
- ❏ テキストファイルの、シーケンシャル書き出しをマスター
- ❏ コンボボックスのリスト項目の設定を行うテクニック
- ❏ ドライブ・ディレクトリ・ファイルの各リストボックスの使い方とファイル名の取得方法
- ❏ コントロールのドラッグ＆ドロップの設定方法

Lesson 22

第5日
1時限目

ファイルを操作する

Lesson22.vbp
VB6CD
Lesson22

◎Learning Edition

テキストファイルを読み込む

テキストファイルを読み込む方法、ループ処理の作り方、および独自のプロシージャを他のプロシージャから呼び出す方法を覚えます。

今回作成するプログラム

1 プログラムを実行すると…

2 「Readme.txt」ファイルの内容が表示される

POINT

このプログラムは、付属CD-ROMのプロジェクトと同じフォルダにある「Readme.txt」というテキストファイルを読み込んで、テキストボックスコントロールに出力するプログラムです。

コントロールとプロパティの設定

※新しいプロジェクトとフォームを用意します。「Readme.txt」ファイルをこのプロジェクトと同じフォルダにコピーしておいてください。

1 フォームの右下隅にあるサイズ変更ハンドル（黒四角）をドラッグしてフォームのサイズを広げる

Form1

フォームのサイズは
Height：4200
Width：7200
を目安にしてください

2 プロパティウィンドウで［BorderStyle］プロパティの値を［1 - 固定（実線）］に設定する

プロパティ - Form1
Form1 Form
全体 | 項目別

(オブジェクト名)	Form1
Appearance	1 - 3D
AutoRedraw	False
BackColor	&H8000000F&
BorderStyle	1 - 固定(実線)
Caption	0 - なし
ClipControls	1 - 固定(実線)
ControlBox	2 - 可変
DrawMode	3 - 固定ダイアログ 4 - 固定ツール ウィンドウ
DrawStyle	5 - 可変ツール ウィンドウ

1 ここをクリック
2 ここをクリック
3 ここをクリック

値の切り替え

［BorderStyle］プロパティのセルをダブルクリックして、「0 - なし」→「1 - 固定（実線）」→「可変」→…というように順に値を切り替えることもできます。

3 ツールボックスの［TextBox］ボタン abl をクリックし、フォーム上にテキストボックスを配置する

Form1
Text1
1 ここをクリックして…
2 そのままここまでドラッグ

[MultiLine] プロパティ

この値を「True」に設定すると、複数行のテキストの取得と表示ができるようになります。

4 プロパティウィンドウで［MultiLine］プロパティの値を「True」に設定する

プロパティー Text1	
Text1 TextBox	
全体	項目別
MouseIcon	(なし)
MousePointer	0 - 既定値
MultiLine	True
OLEDragMode	True
OLEDropMode	False
PasswordChar	
RightToLeft	False

1 ここをクリック
2 ここをクリック
3 ここをクリック

[ScrollBars] プロパティ

この値を「3 - 両方」に設定すると、水平と垂直スクロールバーの両方がテキストボックスに表示されます。

5 ［ScrollBars］プロパティの値を「3 - 両方」に設定する

プロパティー Text1	
Text1 TextBox	
全体	項目別
OLEDropMode	0 - なし
PasswordChar	
RightToLeft	False
ScrollBars	3 - 両方
TabIndex	0 - なし
TabStop	1 - 水平
Tag	2 - 垂直
	3 - 両方
Text	(テキスト)
ToolTipText	
Top	150

1 ここをクリック
2 ここをクリック
3 ここをクリック

6 ツールボックスの［CommandButton］をクリックし、フォーム上にコマンドボタンを配置する

Form1
Text1
Command1

1 ここをクリックして…

2 そのままここまでドラッグ

7 プロパティウィンドウで［Caption］プロパティの値を「終了」に変更する

ﾌﾟﾛﾊﾟﾃｨｰ Command1
Command1 CommandButton
全体　項目別

Appearance	1 - 3D
BackColor	&H8000000F&
Cancel	False
Caption	終了
CausesValidation	True

1 ここをダブルクリック

2 「終了」を入力

8 完成したレイアウト

Form1
Text1
終了

コマンドボタンを選択

この操作を行う前に、作成したコマンドボタンコントロールが選択されていることを確認してください。

コードの記述

Form1:Command1-Click

Project1 - Form1 (コード)

Command1 | Click

```
Private Sub Command1_Click()
    End
End Sub
```

Form1:Form-Load

Project1 - Form1 (コード)

Form | Load

```
Private Sub Form_Load()
    Call FileRead
End Sub
```

Form1:(General)-FileRead

Project1 - Form1 (コード)

(General) | FileRead

```
Private Sub FileRead()
    Dim Buff1 As String
    Dim Buff2 As String

    Open App.Path & "¥Readme.txt" For Input As #1
    Do While Not EOF(1)
        Line Input #1, Buff1
        Buff1 = Buff1 + Chr(13) + Chr(10)
        Buff2 = Buff2 + Buff1
    Loop
    Text1.Text = Buff2
    Close #1
End Sub
```

解　説

1　コードを解説する前に、ファイル操作について説明をしておきます。
ファイルを読み込む場合は、3つの手続きが必要になります。
①ファイルを開く
②ファイルの中身を取り出す
③ファイルを閉じる
最初の「ファイルを開く」というのは、Windowsのファイルシステムに対して、ファイルにアクセスするための処理を行う意味のことをいいます。この操作は、Openステートメントが行います。
2つめの「ファイルの中身を取り出す」という操作は、文字どおりファイルの中のデータを取り出すことをいいます。これは、Input#ステートメントまたはLine Input#ステートメントが担当します。
最後の「ファイルを閉じる」というのは、Openステートメントの逆で、Windowsのファイルシステムに対してファイルへのアクセスを終了することをいいます。これは、Closeステートメントが担当します。

ファイル読み込みの「3つの手続き」（FileRead()プロシージャ）

手続き	ステートメント
①ファイルを開く	「Open」ステートメント
②ファイルの中身を取り出す	「Input#」または「Line Input#」ステートメント
③ファイルを閉じる	「Close」ステートメント

2　ファイルへのアクセスは、ファイル名を使う場合と「ファイル番号」を使う場合の2とおりがあります。ファイル名は、アクセスするファイル名を、ドライブ名も含めたフルパスで用います。
ファイル番号は、Openステートメントでファイルにアクセスしたときに付加する番号で、ファイルを開く順番に1から適用します。

3　では、プログラムのコードの説明に入りましょう。このプログラムは、プログラム起動時に、同じフォルダにある「Readme.txt」というファイルを読み込んで、テキストボックスコントロールに表示します。

プログラム起動時に処理を行うので、FormのLoadイベントプロシージャでファイルの読み込み処理を行います。

ただし、このプログラムでは、別に新しいプロシージャ「FileRead()」を作成して、ここで実際の処理を記述し、FormのLoadイベントプロシージャでは、FileRead()プロシージャを呼び出すだけにします。

4 新しいプロシージャを作成する場合は、コードウィンドウを「モジュール全体を連続表示」に切り替え、行の最下部に「Private Sub FileRead()」とプロシージャ名を入力します。これで、自動的にプロシージャが組み込まれます。

ファイルの読み込み操作は、すべてこのプロシージャで処理します。

```
Project1 - Form1 (コード)
(General)          FileRead

Private Sub Command1_Click()
    End
End Sub

Private Sub Form_Load()
    Call FileRead
End Sub

Private Sub FileRead()

End Sub
```

5 ファイルの読み込み処理は、最初に説明した3つの手続きにそって作成していきます。まず、ファイルの中身を格納するための変数「Buff1」「Buff2」を用意しておきます。

```
Dim Buff1 As String
Dim Buff2 As String
```

次にOpenステートメントを使って、ファイルにアクセスします。

COLUMN ……

ユーザー定義プロシージャ

Visual Basicでは、プログラマが自由にプロシージャを作成できます。その場合は、Subステートメントを使って、コードに記述します。記述するコードは、フォームやコントロールのイベントプロシージャと同じコードウィンドウを使ってもかまいませんし、専用の「標準モジュール」を追加してここに記述してもかまいません。

Subステートメントには、先頭にキーワードが付けられます。「Private」は、同一モジュール内のすべてのプロシージャから参照可能になります。「Public」を付けると、そのプログラムのすべてのモジュールの、すべてのプロシージャから参照できるようになります。プロシージャを作成する場所にあわせて、使い分けてください。

【Openステートメントの書式】

```
Open filename For mode As #fileNo
```

`filename`	アクセスするファイル名を記述します。
`mode`	ファイルを開く際のモードを指定します。 ・`Input`　シーケンシャル入力モード ・`Output`　シーケンシャル出力モード ・`Random`　ランダムアクセスモード ・`Binary`　バイナリモード ・`Append`　シーケンシャル出力の追記モード
`fileNo`	ファイル番号（ファイルを開いた順番）

「Readme.txt」ファイルは、プロジェクトファイルと同じディレクトリにあるので、「App.Path & "¥Readme.txt"」で参照します。［AppPath］プロパティは、プログラム（Appオブジェクト）が実行されているディレクトリまでのフルパスを取得するためのプロパティです（P.154参照）。

そして、このファイルをシーケンシャルに（ファイルの先頭から順番に）読み込む「Input」モードで開きます。最初にファイルを開くので、ファイル番号は「1」になります。

これで、「Readme.txt」ファイルが開かれ、ファイルの中身にアクセスできるようになります。

```
Open App.Path & "¥Readme.txt" For Input As #1
```

6　ファイルを開いたら、中身を取り出します。

ファイルの中身は、Line Input#ステートメントを使って取り出します。Line Input#ステートメントは、1回の実行でファイルの先頭から1行分だけ（改行コードまで）を取り出すステートメントです。

【Line Input# ステートメントの書式】

```
Line Input #fileNo, readbuffer
```

`fileNo`	`Open`ステートメントで実行したときのファイル番号を指定します。
`readbuffer`	読み込んだファイルの中身を格納する変数を指定します。

すでに、Openステートメントでファイル番号を#1に指定して開いていますから、この番号を使います。また、取り出したファイルの中身は、変数Buff1に格納します。これで、一行分のファイルの読み込みが行われます。

```
Line Input #1, Buff1
```

7 何行にもわたるファイルの中身を取り出すには、繰り返しこのステートメントを実行する必要があります。

そこで、Do While...Loopステートメントを使用します。このステートメントは、指定した条件に達するまで、繰り返し処理を実行するループ処理のステートメントです。

【Do While...Loopステートメントの書式】

```
Do While 条件式
        処理...
        処理...
Loop
```

条件式が満足されている間は、Loopまでの間に記述した処理を繰り返し行います。そして、条件式が満足できなくなったら、ループ処理は終了します。

8 このプロシージャでは、ループの条件にEOF関数を使って、ファイルが終端まで達したかどうかをチェックし、ファイルの終端に達していなければ処理を繰り返すように、ループを設定しています。

【EOF関数の書式】

```
EOF(fileNo)
```

EOF関数は、引数にファイル番号を指定すると、そのファイルの読み込みが行われている際に、読み込み位置がファイルの終端に達していると真（True）を返す関数です。

ファイルの読み込み位置は、一回読み込みが行われると、最後に読み込まれた文字の次の位置に移動します。EOF関数を使うと、ファイルの読み込み位置が最後になったかどうかを判断することができますので、これをループの条件に加えれば、ファイルの最後まで読み込みが実行されたかどうかを判断することができます。

そして、Not演算子と組み合わせれば、「ファイルが終端になっていない限り」というループの条件を作れます。

9 改めて、プロシージャのループ処理を整理すると、

①ファイルの読み込み位置がファイルの終端になっていなければループを実行し、Line Input#ステートメントを使って、「Readme.txt」から1行分の文章を読み込みます。

②読み込んだ文章は変数Buff1に格納し、改行文字（Chr(13)+ Chr(10)）を付加して再度変数Buff1に代入しておきます。

③変数Buff2に、Buff1の内容を追加し、再度変数Buff2に代入します。
④もう一度ループの先頭に戻り、EOF関数でファイルの読み込み位置が終端にきていないかどうかをチェックし、終端でなければさらにもう1行分、文章を読み込んで変数に格納する作業を行います。
⑤変数Buff2には、文章が追加される形でどんどん代入されていきます。

ループ処理におけるファイルの読み込み状況（Buff1の場合）


```
Do While Not EOF(1)
    Line Input #1, Buff1
    Buff1 = Buff1 + Chr(13) + Chr(10)
    Buff2 = Buff2 + Buff1
Loop
```

10

1行分を読み込んだところで、改行コードを付加しています。これは、テキストボックスコントロールに文章を表示する際に、きれいに改行して表示するためです。Line Input#ステートメントでは、改行コードは読み込まれず、文字コードしか読み込まれないため、変数に格納する際にわざわざ付加しています。

改行コードを付加するには、Chr関数を使用しています。この関数は、引数に文字のコード番号（ASCIIコードなど）を指定すると、その文字コードに対応する文字を返します。改行コードは通常の文字コードではないので（制御コードと呼ばれています）、ASCIIコードを使って変数に埋め込みます。

ちなみに、WindowsではコードI3（Chr(13)）のキャリッジリターン（CR）とコード10（Chr(10)）のラインフィード（LF）とを組み合わせて改行処理をしているので、この2つを変数に追加しておきます。他の文字についてはP.408の付録「ASCII文字コード一覧」を参照してください。

11

Line Input#ステートメントで読み込みを続け、ファイルの終端まで達すると、Do While...Loopステートメントは、ループの条件が一致しないため、ループ処理を停止します。

変数Buff2には、ループが回った分だけの行数の文章が格納されていますから、そのままBuff2をテキストボックスコントロールの［Text］プロパティに代入します。

テキストボックスコントロールは、［MultiLine］プロパティをTrueに設定しました。これで、テキストボックスは複数行にわたる文章を操作できるようになっています。また、［ScrollBars］プロパティも、縦横のスクロールバーが表示されるように設定してありますので、［Text］プロパティに代入した内容がテキストボックスコントロールのサイズを越えている場合は、自動的にスクロールバーがテキストボックスに表示されるようになります。

```
Text1.Text = Buff2
```

12 最後に、ファイルからすべての読み出しが終了した時点で、Closeステートメントを使ってファイルを閉じておきます。Closeステートメントは、ファイル番号を指定するだけで、Windowsのファイルシステムに対し、ファイルを閉じる処理を行ってくれます。

```
Close #1
```

13 以上で、ファイルの読み込み処理を行うプロシージャが完成しました。後は、フォームのLoadイベントプロシージャでこのプロシージャを呼び出すだけです。

プロシージャで別のプロシージャを呼び出すには、Callステートメントを使用します。このとき、呼び出すプロシージャは（）をはずしたプロシージャ名だけを記述します。

```
Private Sub Form_Load()
    Call FileRead
End Sub
```

14 そして、コマンドボタンにプログラムの終了処理であるEndステートメントを記述して完成です。プログラムを起動すると、同じフォルダにある「Readme.txt」ファイルが読み込まれて、テキストボックスコントロールに表示されます。

COLUMN ・・・・・

ループ処理

1つの処理を何回も繰り返し実行することを「ループ処理」といいます。Visual Basicでは、ループの形態でいくつかのステートメントが用意されています。「Do While Loop」は、条件を指定するとその条件が満足されている間、指定した処理を繰り返します。何回ループがまわるかは、設定した条件に依存します。「For Next」ステートメントは、指定した回数だけループをまわします。あらかじめ繰り返したい回数が決まっている場合は、このステートメントを使ったほうが便利です。

まとめ

文章で説明すると、かなり長い説明になってしまいますが、処理はさほど難しくありません。ファイル操作の「3つの手続き」に沿って処理を組み立てていけば、混乱することなく作成できるでしょう。そして、ファイルを開いたら、必ずCloseステートメントで閉じておくことを忘れないでください。ファイルを閉じないとファイルを破壊する原因になってしまいます。また、今回のプログラムは、あくまでもテキストファイルが操作対象です。プログラムファイルなどのバイナリファイルは開かないでください。これも、ファイルを破壊する原因になります。

練習問題

Q1　今回、ファイルから文章を取り出すのに、Line Input# ステートメントを用いました。しかし、ファイルから文章を取り出すステートメントは、他にもあります。

Input# ステートメントは、同じようにテキストファイルから文字を読み込むステートメントですが、カンマや空白（スペース）は無視して変数に格納します。「Lesson22.vbp」をInput# ステートメントを用いて処理するように改修し、Line Input# ステートメントとの違いを確認してください。また、Input# ステートメントの使い方を、Visual Basicのヘルプファイルを使って調べてください。

Line Input# ステートメントを使った場合

Input# ステートメントを使うと空白が飛ばされる

解答は巻末に→

ワンポイントアドバイス

ファイルの内容を表示するのに、テキストボックスコントロールを使用しました。文章のボリュームが大きいファイルの場合は、ラベルコントロールよりもテキストボックスコントロールのほうが、複数行の表示ができることと、スクロールバーが付加されるので表示操作が格段に優れているためです。ただし、これらの機能を使う場合は、必ず［MultiLine］と［ScrollBars］プロパティを設定してください。

Lesson 23

第5日
2時限目

ファイルを操作する

Lesson23.vbp
VB6CD
Lesson23

◎Learning Edition

リスト項目をファイルに書き出す

テキストファイルの、シーケンシャル書き出しをマスターします。また、コンボボックスのリスト項目の設定を行うテクニックを覚えます。

● 今回作成するプログラム

ガールフレンド名簿
美穂
選択 追加 削除
Lesson23
リストボックスに追加しますか？
はい(Y) いいえ(N)
List.dat - メモ帳
ファイル(F) 編集(E) 検索(S) ヘルプ(H)
麻美
留美子
ジェニー
あみ
なぎさ
美穂
ガールフレンド名簿
美穂
麻美
留美子
ジェニー
あみ
なぎさ
美穂
終了

1 リストに名前を追加すると…

2 List.dat ファイルに名前が保存され…

3 次回もリストに名前が表示される

POINT

「Lesson21.vbp」で作成したコンボボックスのプログラムは、リストボックス部分の項目名を追加しても保存できませんでした。そこで、専用のデータファイルを作成し、プログラム終了時にこのデータファイルにリスト項目名を書き出すプログラムを作成します。

また、ファイルの読み込み処理の復習をかねて、プログラム起動時にリスト項目を初期化する機能を組み込みます。

コントロールとプロパティの設定

1 「Lesson21.vbp」を開く

ガールフレンド名簿
彼女の名前を入力して下さい。
選択　追加　削除

2 フォームを下方向に広げる

ガールフレンド名簿
彼女の名前を入力して下さい。
選択　追加　削除

Height ： 4100 程度

1 サイズ変更ハンドルを下方向にドラッグ

3 ツールボックスの［CommandButton］をクリックし、フォーム上にコマンドボタンを配置する

Command1

1 ここをクリックして…

2 そのままここまでドラッグ

Lesson21.vbp の保存場所

このファイルは、付録CD-ROMの「Lesson21」フォルダの中に収録されています。

4 プロパティウィンドウで［Caption］プロパティの値を「終了」に変更する

プロパティ - ExitBtn	
ExitBtn CommandButton	
(オブジェクト名)	ExitBtn
Appearance	1 - 3D
BackColor	&H8000000F&
Cancel	False
Caption	終了
CausesValidation	True
Default	False

1 ここをダブルクリック

2 「終了」と入力

5 ［(オブジェクト名)］プロパティの値を「ExitBtn」に変更する

プロパティ - ExitBtn	
ExitBtn CommandButton	
(オブジェクト名)	ExitBtn
Appearance	1 - 3D
BackColor	&H8000000F&
Cancel	False
Caption	終了
CausesValidation	True
Default	False

1 ここをダブルクリック

2 「ExitBtn」と入力

6 完成したレイアウト

ガールフレンド名簿

彼女の名前を入力して下さい。

選択　追加　削除

終了

［List］プロパティの消去

コンボボックスの［List］プロパティにあるリスト項目は消去しておいてください。

コードの記述

Form1:(General)-FileSave

Project1 - Form1 (ｺｰﾄﾞ)

(General) | FileSave

```
Private Sub FileSave()
    Dim WriteData As String
    Dim i As Integer

    i = 0
    Open App.Path & "¥List.dat" For Output As #1
    Do While i < Combo1.ListCount
        WriteData = Combo1.List(i)
        i = i + 1
        Print #1, WriteData
    Loop
    Close #1
End Sub
```

Form1:(General)-FileRead

Project1 - Form1 (ｺｰﾄﾞ)

(General) | FileRead

```
Private Sub FileRead()
    Dim ItemName As String

    Open App.Path & "¥List.dat" For Input As #1
    Do While Not EOF(1)
        Line Input #1, ItemName
        Combo1.AddItem ItemName
    Loop
    Close #1
End Sub
```

Form1:ExitBtn-Click

Project1 - Form1 (ｺｰﾄﾞ)

ExitBtn | Click

```
Private Sub ExitBtn_Click()
    Call FileSave
    End
End Sub
```

Form1:Form-Load

Project1 - Form1 (ｺｰﾄﾞ)

Form | Load

```
Private Sub Form_Load()
    Call FileRead
End Sub
```

Form1:Form-Unload

Project1 - Form1 (ｺｰﾄﾞ)

Form | Unload

```
Private Sub Form_Unload(Cancel As Integer)
    Call FileSave
End Sub
```

COLUMN ・・・・・

ファイル操作のステートメント・関数一覧

ファイルを操作するには、いくつかのステートメントと関数を使用します。ここでは、代表的なものを紹介します。

＜ファイル操作のステートメント＞

ステートメント	説明
Open	ファイルを開き、内容にアクセスできるようにします。
Close	開いたファイルを閉じます。
Print#	シーケンシャル出力モード（Output または Append）で開いたファイルにデータを書き込みます。
Write#	シーケンシャル出力モード（Output または Append）で開いたファイルにデータを書き込みます。

＜ファイル操作の関数＞

関数	説明
FreeFile	使用可能なファイル番号を返すファイル入出力関数です。
FileLen	ファイルのサイズをバイト単位で返します。
FileDateTime	ファイルの作成日時または最後に修正した日時を返します。
Input	シーケンシャル入力モード（Input）またはバイナリモード（Binary）で開いたファイルから指定した文字数の文字列を読み込み返します。
LOF	Open ステートメントを使用して開いたファイルの長さをバイト単位で返します。

解 説

1 今度は、テキストファイルのシーケンシャル書き出し処理を行います。ファイルの書き出しも、「3つの手続き」（P.203参照）で処理を組み立てることには変わりありません。ファイルの読み込み部分が書き出しに入れ替わるだけです。
プログラムは、終了時にコンボボックスのリスト項目を、「List.dat」という専用のデータファイルを作成して、そこに書き出します。この処理は、「FileSave」というプロシージャを作成して行います。

2 プロシージャFileSaveは、Do Whileループを使って、リスト項目を順番にファイルに書き込んでいきます。
まず、書き込む項目名を格納する変数と、ループの回数を数えるカウンタ用の変数を宣言します。カウンタ用変数「i」は、値を0に初期化しておきます。

```
Private Sub FileSave()
    Dim WriteData As String
    Dim i As Integer

    i = 0
```

3 データを書き込むファイル「List.dat」は、プロジェクトファイルと同じフォルダ内に置きます。また、Openステートメントは書き込み（出力）モード（Output）で実行します。Openステートメントを書き込みモードで実行したときは、存在しないファイル名を指定した場合でも、自動的にそのファイルが作成されます。

```
Open App.Path & "¥List.dat" For Output As #1
```

Openステートメントの書式はP.205を参照してください。

4 ファイルにアクセスできたら、データを書き込んでいきます。
リスト項目の総数をコンボボックスの［ListCount］プロパティを使って調べ、その数だけDo Whileループを回します。

```
Do While i < Combo1.ListCount
    処理
Loop
```

そしてコンボボックスの［List］プロパティを使って、リスト項目を1つずつ参照していきます。たとえば、

```
WriteData = Combo1.List(0)
```

とすると、リスト項目の先頭（「0」番目）の項目名を取得して、変数WriteDataに格納できます。

このプロパティを使って、リストを「0」からListCount分（リスト項目の総数）まで1つずつ参照していけばいいわけです。リスト項目を1つ把握したら、カウンタ用変数「i」を1つ増加させて、次のループで次のリスト項目が参照できるようにします。

```
WriteData = Combo1.List(i)
i = i + 1
```

5 ファイルにデータを書き込むには、Print#ステートメントを使います。

```
Print #1, WriteData
```

【Print# ステートメントの書式】

```
Print #fileNo, data
```

fileNo	書き込むファイル番号を指定します。通常は、Openステートメントで使ったファイル番号を使います。
data	書き込むデータを指定します。

これで、「List.dat」ファイルにリストの項目名が書き込まれます。

6 ループの条件であるカウンタ用変数の値がリスト項目の総数未満に達すると（i < Combo1.ListCount）、ループは停止します。カウンタ変数は「0」から始まっていますから、リストの総項目数「-1」の値でループが停止します。

ループによる、データのファイルへの書き込みが終われば、Closeステートメントでファイルを閉じます。

書き込みの場合も、必ずこの処理を行ってください。

```
Loop
Close #1
```

7 ファイルの書き込み処理が完成したら、このFileSaveプロシージャを呼ぶCallステートメントを、フォームの終了プロシージャに記述します。フォームが終了するときは、Unloadイベントプロシージャが発生しますので、ここでFileSaveプロシージャを呼び出します。

```
Private Sub Form_Unload(Cancel As Integer)
    Call FileSave
End Sub
```

また、配置した［終了］ボタンのClickイベントプロシージャにも、Endステートメントを実行する前にFileSaveプロシージャを呼び出しておきます。

これで、プログラム終了時は、必ず最後のリスト項目がデータファイルに保存されます。

```
Private Sub ExitBtn_Click()
    Call FileSave
    End
End Sub
```

8　データファイルが作成されたら、次回プログラム起動時に、このファイルを読み込んで、コンボボックスのリスト項目を設定するようにしましょう。これも、FileReadプロシージャを作成して、ここにその処理を記述します。

ファイル読み込みの処理は、Lesson22のプログラムと同様に、Line Input#ステートメントを使って、1行ずつ読み込んでいきます。そして、読み込んだデータを、AddItemメソッドを使ってコンボボックスに設定していきます。

この処理は、Do WhileループとEOF関数を用いて、ファイルの最後まで繰り返されます。これで、ファイルに保存したすべてのリスト項目がコンボボックスに設定されます。

```
Private Sub FileRead()
    Dim ItemName As String

    Open App.Path & "¥List.dat" For Input As #1
    Do While Not EOF(1)
        Line Input #1, ItemName
        Combo1.AddItem ItemName
    Loop
    Close #1
End Sub
```

9　ファイルの読み込みプロシージャ「FileRead」は、プログラム起動時に実行させますから、フォームのLoadイベントプロシージャで、Callステートメントを使って呼び出します。

これで、前回終了時に設定されていたリスト項目がすべてコンボボックスに設定されます。

```
Private Sub Form_Load()
    Call FileRead
End Sub
```

10 以上で、完成です。プログラムを動作させて、リスト項目の追加や削除を実行し、［終了］ボタンでプログラムを終了させます。Windowsのメモ帳を起動して、「List.dat」を調べてください。きちんとリスト項目が記述されています。

再度、プログラムを起動すると、「List.dat」の中身がコンボボックスのリストに設定されます。

ガールフレンド名簿
真美
麻美
留美子
ジェニー
あみ
なぎさ
美穂
真美
終了

設定したリスト項目を保存

List.dat - メモ帳
ファイル(F) 編集(E) 検索(S) ヘルプ(H)
麻美
留美子
ジェニー
あみ
なぎさ
美穂
真美

保存されたList.datの中身

ガールフレンド名簿
彼女の名前を入力して下さい。
麻美
留美子
ジェニー
あみ
なぎさ
美穂
真美
終了

きちんとリストに設定されている

まとめ

　テキストデータのファイルへの書き出しは、データによっては改行コードを付加したほうがいいものもあります。
　また、Write# ステートメントを使うと、データを"（引用符）で囲んでファイルに保存することができますから、Excel などでデータを再利用する場合などは、このステートメントを使うのも一つの手段です。
　特定の処理を１つのプロシージャにまとめて、Call ステートメントで呼び出す方法は、複数のプロシージャから参照する場合などは、同じ記述をいくつも作らなくて済むので便利です。また、そのプロシージャだけをコピーして他のプログラムでも再利用できますから、このようなプロシージャの作り方をする習慣をつけておくといいでしょう。

練習問題

Q1　リスト項目のファイルへの書き出しの際に、メッセージボックスを使って保存するかどうかを選べるように、「Lesson23.vbp」を修正してください。

【ヒント】
　MsgBox 関数にボタンを設定する方法はすでに説明しましたね（P.192 参照）。これと If ステートメントを組み合わせて、［はい］ボタンが押されたときだけ、ファイルにリスト項目を書き込むようにします。

ガールフレンド名簿
彼女の名前を入力してください。
選択　追加　削除
データの保存
リスト項目を保存しますか
はい(Y)　いいえ(N)　キャンセル
終了

プログラム実行画面

解答は巻末に→

Lesson 24

第５日
３時限目

ファイルを操作する

Lesson24.vbp
VB6CD
Lesson24

◎Learning Edition

イメージビューワーの作成

ドライブ、ディレクトリ、ファイルの各リストボックスの使い方をマスターし、ファイル名を取得して画像を表示する方法を覚えます。

● 今回作成するプログラム

1 ドライブとフォルダを指定し…

2 画像ファイルを選択すると…

ドライブリストボックス

3 選択した画像が表示される

ディレクトリリストボックス

ファイルリストボックス

!!! POINT

ドライブ、ディレクトリ、およびファイルリストボックスを組み合わせて、任意のドライブのイメージファイルにアクセスし、ファイル名を取り出します。取り出したファイル名は、イメージコントロールに渡し、そのイメージを表示します。また、ファイルリストボックスでは、OLEを使ったドラッグ＆ドロップの設定を行い、ファイルリストボックスのファイルをデスクトップやフォルダ、アプリケーションにドラッグ＆ドロップできるようにします。

コントロールとプロパティの設定

※新しいプロジェクトとフォームを用意します。

1 フォームのサイズを広げる

Project1 - Form1 (Form)

Form1

Heights : 5000
Width : 7000

1 サイズ変更ハンドルをドラッグ

2 ツールボックスの［DriveListBox］をクリックし、フォーム上にドライブリストボックスを配置する

1 ここをクリックして…

Form1

i: [SDK & DEV60]

2 そのままここまでドラッグ

フォームのサイズを広げる

フォームの右下隅にあるサイズ変更ハンドルを外側に向かってドラッグします。

3 ツールボックスの［DirListBox］をクリックし、フォーム上にディレクトリリストボックスを配置する

Form1

i: [SDK & DEV60]

I:¥
Program Files
VB98
Template
Tsql
Wizards

1 ここをクリックして…

2 そのままここまでドラッグ

4 ツールボックスの［FileListBox］をクリックし、フォーム上にファイルリストボックスを配置する

Form1

i: [SDK & DEV60]

I:¥
Program Files
VB98
Template
Tsql
Wizards

ADDSCCJP.DLL
ADDSCCUS.DLL
BIBLIO.MDB
C2.EXE
CVPACK.EXE
DATAVIEW.DLL
LINK.EXE
MSDIS110.DLL
MSPDB60.DLL
NWIND.MDB

1 ここをクリックして…

2 そのままここまでドラッグ

ファイルリストボックスを選択

この操作を行う前に、作成したファイルリストボックスコントロールが選択されていることを確認してください。

5 ファイルリストボックスの［OLEDragMode］プロパティの値を「1 - 自動」に変更する

File1 FileListBox

全体 | 項目別

MousePointer	0 - 既定値
MultiSelect	0 - なし
Normal	True
OLEDragMode	1 - 自動
OLEDropMode	0 - なし

1 セルをダブルクリック

2 値が切り替わる

6 [Pattern] プロパティの値を「*.bmp;*.jpg;*.gif」に変更する

プロパティ - File1
File1 FileListBox
全体 | 項目別

OLEDragMode	1 - 自動
OLEDropMode	0 - なし
Pattern	*.bmp;*.jpg;*.gif
ReadOnly	True
System	False
TabIndex	2
TabStop	True

1 ここをダブルクリック
2 値を入力

7 ツールボックスの [Image] ボタンをクリックし、フォーム上にイメージコントロールを配置する

Form1
c: [SDK & DEV60]
c:¥
Program Files
VB98
Template
Tsql
Wizards

1 ここをクリックして…
2 そのままここまでドラッグ

8 プロパティウィンドウで [BorderStyle] プロパティの値を「1 - 実線」に設定する

プロパティ - Image1
Image1 Image
全体 | 項目別

(オブジェクト名)	Image1
Appearance	1 - 3D
BorderStyle	1 - 実線
DataField	0 - なし
DataFormat	1 - 実線
DataMember	
DataSource	

1 ここをクリック
2 ここをクリック
3 ここをクリック

パターン

[Pattern] プロパティを設定すると、指定された拡張子（「bmp」「jpg」など）の種類のファイルのみをファイルリストボックス内に表示させるようにすることができます。ワイルドカード「*.*」を指定することもできます。

Windowsアプリケーションでの [ファイルを開く] ダイアログボックスの中にある [ファイルの種類] を使う方法と似ています。

イメージを選択

この操作を行う前に、作成したイメージコントロールが選択されていることを確認してください。

[BorderStyle] プロパティをダブルクリックして、[0 - なし] と [1 - 実線] を切り替えることもできます。

9 ツールボックスの［CommandButton］ をクリックし、フォーム上にコマンドボタンを配置する

Template
Tsql
Wizards
Command1

1 ここをクリックして…

2 そのままここまでドラッグ

10 プロパティウィンドウで［Caption］プロパティの値を「終了」に変更する

Command1 CommandButton

全体 | 項目別

(オブジェクト名)	Command1
Appearance	1 - 3D
BackColor	&H8000000F&
Cancel	False
Caption	終了
CausesValidation	True

1 ここをダブルクリック

2 「終了」を入力

コードの記述

Form1:Drive1-Change

Project1 - Form1 (コード)

Drive1 | Change

```
Private Sub Drive1_Change()
    Dir1.Path = Drive1.Drive
End Sub
```

Form1:Dir1-Change

Project1 - Form1 (コード)

Dir1 | Change

```
Private Sub Dir1_Change()
    File1.Path = Dir1.Path
```

Form1:File1-Click

Project1 - Form1 (コード)

File1 | Click

```
Private Sub File1_Click()
    Dim Fname As String

    Fname = Dir1.Path & "¥" & File1.filename
    Image1.Picture = LoadPicture(Fname)
```

Form1:Command1-Click

Project1 - Form1 (コード)

Command1 | Click

```
Private Sub Command1_Click()
    End
End Sub
```

解 説

1 ファイルにアクセスするには、ドライブリストボックス、ディレクトリリストボックス、ファイルリストボックスの、3つのコントロールを連動させるのがポイントです。

Form1

f: [WEB]

f:¥ / _seto_www / book / cg / cute / New

about-bk.gif / about-bk.jpg / anistar1.gif / back-4.gif / bk.jpg / b-line6.gif / b-line7.gif / block-2.gif

ドライブリストボックスは、コンピュータで利用可能なドライブをリスト表示します。

ディレクトリリストボックスは、指定したドライブのディレクトリ（フォルダ）名を一覧表示します。

ファイルリストボックスは、指定したディレクトリ（フォルダ）にあるファイルを一覧表示します。

2 ドライブリストボックス（Drive1）は、ユーザーによってドライブが選択されると、Changeイベントが発生します。そして、選択されたドライブ名は［Drive］プロパティに格納されます。

この［Drive］プロパティの値を、ディレクトリリストボックス（Dir1）の［Path］プロパティに代入すると、ディレクトリリストボックスは、そのドライブのディレクトリ一覧にリストを更新します。そこで、ドライブボックスのChangeイベントプロシージャに、この処理を記述します。

```
Private Sub Drive1_Change()
    Dir1.Path = Drive1.Drive
End Sub
```


3 次に、ディレクトリリストボックスは、ユーザーによってディレクトリ（フォルダ）が選択されると、Changeイベントを発生します。選択されたディレクトリ（フォルダ）名は、［Path］プロパティに格納されます。

この［Path］プロパティの値（Dirl.Path）を、ファイルリストボックス（File1）の［Path］プロパティ（File1.Path）に代入すると、今度はファイルリストボックスがこのディレクトリ（フォルダ）のファイル一覧に更新されます。

```
Private Sub Dir1_Change()
    File1.Path = Dir1.Path
End Sub
```

4 ファイルリストボックスは、ユーザーによってファイルが選択されると、そのファイル名を［filename］プロパティに格納します。この値を取り出して、イメージコントロールの［Picture］プロパティに代入すれば、そのイメージファイルを表示できます。

ここでは、ファイルをクリックするだけで、イメージコントロールに表示するように、ファイルリストボックスのClickイベントプロシージャに処理を記述します。

5 イメージコントロールにイメージファイルを表示するには、［Picture］プロパティに「LoadPicture」メソッドを使ってイメージファイルを代入します。

直接［Picture］プロパティにファイル名を代入しても表示されません。LoadPictureメソッドを使って、イメージファイルを読み込んで、メモリに格納します。それを、イメージコントロールに渡せばイメージが表示されます。

```
Image1.Picture = LoadPicture(ファイル名)
```

6

LoadPictureメソッドは、引数にファイル名をフルパスで記述します。
しかし、ファイルリストボックスの［filename］プロパティに格納されるのはファイル名のみで、ドライブやディレクトリ（フォルダ）名は格納されません。そこで、ディレクトリリストボックスの［Path］プロパティと組み合わせて、フルパスのファイル名を作成してあげます。このとき「¥」記号を忘れず追加しておきます。なお、ファイルリストボックスの［Path］プロパティには、ドライブ名も含まれて格納されるので「Dir1.Path」とするところを「File1.Path」としてもかまいません。

```
Private Sub File1_Click()
    Dim Fname As String

    Fname = Dir1.Path & "¥" & File1.filename
    Image1.Picture = LoadPicture(Fname)
End Sub
```

7

［終了］ボタンには、プログラムの終了処理を作成します。

```
Private Sub Command1_Click()
    End
End Sub
```

8

ドライブ、ディレクトリ、ファイルの各リストボックスは、［Drive］［Path］プロパティが変更されると、自動的にChangeイベントが発生します。コードでプロパティの値を変えても、発生します。これによって、3つのリストボックスが連動していれば、ドライブを変更するだけで自動的にディレクトリとファイルの一覧が更新されます。

9

ファイルリストボックスの［Pattern］プロパティは、リストに表示するファイルの種類を指定するプロパティです。ここでは、「BMP」「JPEG」「GIF」の3種類のファイルだけ表示するように、［Pattern］プロパティを設定しました。
複数のパターンを設定する場合は、「;」で区切って記述します。

```
*.bmp;*.jpg;*.gif
```

ワイルドカードが使えますので、目的に応じたパターンを設定してください。

d:¥
_GENKOU
10ka-VB6-1
DHTML
画面
原稿

LESSON00.TXT
LESSON01.TXT
LESSON02.TXT
LESSON03.TXT
LESSON04.TXT
LESSON05.TXT
LESSON06.TXT
LESSON07.TXT

Patternを「*.txt」に変えて実行した

10 ファイルリストボックスの、［OLEDragMode］プロパティを「1 - 自動」に設定すると、選択したファイルをデスクトップやフォルダにドラッグ＆ドロップできるようになります。また、アプリケーションがOLEサーバに対応していれば、ファイルリストボックスからファイルをドラッグ＆ドロップで渡すことができます。

11 以上で完成です。プログラムを実行します。

ドライブリストボックスでドライブを変更すると、自動的にディレクトリとファイルリストボックスの表示が更新されます。また、ファイルをクリックすると、イメージコントロールに表示されます。フォームは、［BorderStyle］プロパティを「2 - 可変」のままにしてありますので、表示しきれないイメージの場合は、フォームを広げて表示できます。

ファイルリストボックスのドラッグ＆ドロップ機能により、ファイルをドラッグ＆ドロップして使うことができます。

選択したファイルのイメージを表示するためにフォームを広げることができる

1 ファイルリストボックスからファイルをドラッグして…

2 別のソフトにドラッグ＆ドロップできる

まとめ

ファイル関係のリストボックスを使うと、マウスだけでファイルの操作が実行できます。Windows には、[ファイルを開く] ダイアログボックスなどのコモンダイアログボックスがありますが、ファイルリストを表示したまま処理を行うには、これらのファイル操作のリストボックスを組み合わせて使うと便利です。

ポイントは、ドライブ・ディレクトリリストボックスでユーザーがクリックすると、Click イベントではなく Change イベントが発生することです。

また、コードでドライブやディレクトリを変更しても、Change イベントが自動的に発生することです。これらのコントロールの、動作の特徴を把握しておきましょう。

練習問題

Q1 作成した「Lesson24.vbp」を修正して、ファイルリストボックスの Pattern プロパティの設定を、コンボボックスで行えるようにしてください。コンボボックスは、以下の３つの Pattern を選べるようにします。

*.bmp　　*.jpg　　*.gif

Form1
d:
D:¥
Program Files
Microsoft Visual Studi
VB98
Setup
Template
*.bmp
*.bmp
*.jpg
*.gif
終了

【ヒント】

コンボボックスで選択された項目は、[Text] プロパティに格納されます。これを、ファイルリストボックスの [Pattern] プロパティに代入すれば、指定したファイルの拡張子だけを表示できます。

解答は巻末に→

ワンポイントアドバイス

ディスクからファイルを削除するには、「Kill」ステートメントを使用します。ディレクトリ（フォルダ）を作成するには、「MkDir」ステートメントを、ディレクトリを削除するには、「RmDir」ステートメントを使用します。

カレントのディレクトリ（現在作業中のディレクトリ）を変更するには、ChDir ステートメントを使います。

＜Kill ステートメントの例＞

①１つのファイルを削除する場合

```
Kill "Readme.txt"
```

②現在のフォルダにあるすべての *.TXT ファイルを削除する場合

```
Kill "*.TXT"
```

＜MkDir ステートメントの例＞

①ドライブ C に「MyData」というディレクトリを作成する場合

```
MkDir "MYData"
```

＜ChDir ステートメントの例＞

①１つ上の階層に移動する場合

```
ChDir "..¥"
```

Let's VBSchool!!

第6日 (Lesson25 (1) ～ Lesson25 (2))

ソフトウェアランチャーの作成

Windowsのプログラムでは、プログラムに使用する各種設定をファイルに保存できます。

Windows 3.1のときは、「.ini」ファイル（プロファイルファイル）にこの設定を保存しましたが、Windows 95からは「レジストリ」に格納するようになりました。

このレッスンでは、これまで紹介してきたテクニックとあわせ、レジストリを操作するテクニックを使い、複数のアプリケーションを一度に起動する、「ソフトウェアランチャー」を作成します。

作成するプログラムは比較的長くやや複雑になるので、機能別に2つのフォームに分けて、各フォームごとに作成していきます。

今日のレッスンで解説する項目

- ❏ Windowsのレジストリの読み書きを行うステートメントの使い方
- ❏ Windowsのレジストリの仕組み
- ❏ ファイル関係のリストボックスの使い方
- ❏ 2つのフォーム間のデータの操作
- ❏ 他のアプリケーションプログラムを実行する方法

Lesson 25(1)

第６日 １時限目

ソフトウェアランチャーの作成

Lesson25.vbp
VB6CD
Lesson25

◎Learning Edition

メインモジュールの作成

プログラムのメインモジュールを作成します。また、Windowsのレジストリの読み書きを行うステートメントの使い方を覚えます。

● 今回作成するプログラム

プログラムランチャー
登録済みアプリケーション
c:¥WINNT¥SYSTEM32¥CALC.EXE
追加
削除
Run
終了

1 登録したプログラムを選択して…

2 ［Run］ボタンをクリックすると…

電卓
編集(E) 電卓の種類(V) ヘルプ(H)
Back CE C
MC 7 8 9 / sqrt
MR 4 5 6 * %
MS 1 2 3 - 1/x
M+ 0 +/- . + =

3 プログラムが起動する

POINT

複数のソフトウェアを、一度に起動するプログラムです。起動するソフトは、専用のダイアログボックスで登録します。登録したソフトは、Windows 95/98のレジストリファイル（レジストリデータベース）に登録し、プログラム起動時に読み込んで、実行します。

また、登録ソフトは、いつでも自由に変更できるようにします。

解説の便宜上、登録できるソフトは、４つまでとします。

コントロールとプロパティの設定

※新しいプロジェクトとフォームを用意します。

1 ツールボックスの［Label］ をクリックし、フォーム上にラベルコントロールを配置する

1 ここをクリックして…

2 そのままここまでドラッグ

2 プロパティウィンドウで［Caption］プロパティの値を「登録済みアプリケーション」に変更する

AutoSize	False
BackColor	&H8000000F&
BackStyle	1 - 不透明
BorderStyle	0 - なし
Caption	登録済みアプリケーション

1 ここをダブルクリック

2 値を入力

3 ［BorderStyle］プロパティの値を「1 - 実線」に設定する

AutoSize	False
BackColor	&H8000000F&
BackStyle	1 - 不透明
BorderStyle	1 - 実線
Caption	0 - なし / 1 - 実線
DataField	
DataFormat	

1 ここをクリック

2 ここをクリック

3 ここをクリック

次のレッスンで完成

このレッスンは2時限目まで作成した時点で完成です。1時限目までを作って、プログラムを実行するとエラーが出るので、注意してください。

ラベルを選択

この操作を行う前に、作成したラベルコントロールが選択されていることを確認してください。

値の切り替え

［BorderStyle］プロパティをダブルクリックして、［0 - なし］と［1 - 実線］を切り替えることもできます。

4 [Alignment] プロパティの値を「2 - 中央揃え」に設定する

ﾌﾟﾛﾊﾟﾃｨｰ Label1
Label1 Label
全体 | 項目別

(ｵﾌﾞｼﾞｪｸﾄ名)	Label1
Alignment	2 - 中央揃え
Appearance	0 - 左揃え
AutoSize	1 - 右揃え
BackColor	2 - 中央揃え
BackStyle	1 - 不透明
BorderStyle	1 - 実線

1 ここをクリック
2 ここをクリック
3 ここをクリック

5 ツールボックスの [ListBox] をクリックし、フォーム上にリストボックスコントロールを配置する

Form1
登録済みアプリケーション
List1

1 ここをクリックして…
2 そのままここまでドラッグ

6 プロパティウィンドウで [ToolTipText] プロパティに「クリックして選択」と入力する

ﾌﾟﾛﾊﾟﾃｨｰ List1
List1 ListBox
全体 | 項目別

TabIndex	1
TabStop	True
Tag	
ToolTipText	クリックして選択
Top	675
Visible	True
WhatsThisHelpID	0

1 ここをダブルクリック
2 値を入力

リストボックスを選択

この操作を行う前に、作成したリストボックスコントロールが選択されていることを確認してください。

7 [Sorted] プロパティの値を「True」に設定する

プロパティ - List1	
List1 ListBox	
MultiSelect	0 - なし
OLEDragMode	0 - 手動
OLEDropMode	0 - なし
RightToLeft	False
Sorted	True
Style	0 - 標準

1 セルをダブルクリック

8 ツールボックスの [CommandButton] をクリックし、フォーム上にコマンドボタンを配置する

Form1
登録済みアプリケーション
List1
Command1

1 ここをクリックして…

2 そのままここまでドラッグ

9 プロパティウィンドウで [Caption] プロパティの値を「追加」に変更する

プロパティ - Command1	
Command1 CommandButton	
(オブジェクト名)	Command1
Appearance	1 - 3D
BackColor	&H8000000F&
Cancel	False
Caption	追加
CausesValidation	True

1 ここをダブルクリック

2 値を入力

[Sorted]プロパティ

このプロパティを「True」に設定すると、リストボックスの項目が自動的にアルファベット順に並べ替えられます。

コマンドボタンを選択

この操作を行う前に、作成したコマンドボタンコントロールが選択されていることを確認してください。

10 [(オブジェクト名)] プロパティの値を「AppAdd」に変更する

AppAdd CommandButton	
(オブジェクト名)	AppAdd
Appearance	1 - 3D

1 ここをダブルクリック
2 値を入力

11 [ToolTipText] プロパティに「登録アプリケーションの追加」と入力する

AppAdd CommandButton	
TabStop	True
Tag	
ToolTipText	登録アプリケーションの追加
Top	2040

1 ここをダブルクリック
2 値を入力

12 ツールボックスの [CommandButton] をクリックし、フォーム上にコマンドボタンを配置する

Form1
登録済みアプリケーション
List1
追加
Command1

1 ここをクリックして…
2 そのままここまでドラッグ

13 プロパティウィンドウで [Caption] プロパティの値を「削除」に変更する

Command1 CommandButton	
BackColor	&H8000000F&
Cancel	False
Caption	削除
CausesValidation	True
Default	False

1 ここをダブルクリック
2 値を入力

コマンドボタンを選択

この操作を行う前に、作成したコマンドボタンコントロールが選択されていることを確認してください。

14 [(オブジェクト名)] プロパティの値を「AppDel」に変更する

AppDel CommandButton
全体 | 項目別

(オブジェクト名)	AppDel
Appearance	1 - 3D
BackColor	&H8000000F&
Cancel	False

1 ここをダブルクリック
2 値を入力

15 [ToolTipText] プロパティに「登録アプリケーションの削除」と入力する

AppDel CommandButton
全体 | 項目別

TabStop	True
Tag	
ToolTipText	登録アプリケーションの削除
Top	2070
UseMaskColor	False
Visible	True

1 ここをダブルクリック
2 値を入力

ツールチップ

ツールチップについてはLesson19を参照してください。

16 ツールボックスの [CommandButton] をクリックし、フォーム上にコマンドボタンを配置する

追加　削除　Command1

1 ここをクリックして…
2 そのままここまでドラッグ

17 プロパティウィンドウで [Caption] プロパティの値を「Run」に変更する

Command1 CommandButton
全体 | 項目別

BackColor	&H8000000F&
Cancel	False
Caption	Run
CausesValidation	True
Default	False
DisabledPicture	(なし)

1 ここをダブルクリック
2 値を入力

コマンドボタンを選択

この操作を行う前に、作成したコマンドボタンコントロールが選択されていることを確認してください。

18 [(オブジェクト名)] プロパティの値を「Run」に変更する

プロパティ - Run	
(オブジェクト名)	Run
Appearance	1 - 3D
BackColor	&H8000000F&
Cancel	False
Caption	Run
CausesValidation	True
Default	False

1 ここをダブルクリック
2 値を入力

19 [ToolTipText] プロパティに「アプリケーションの実行」と入力する

プロパティ - Run	
TabStop	True
Tag	
ToolTipText	アプリケーションの実行
Top	3015
UseMaskColor	False
Visible	True
WhatsThisHelpID	0

1 ここをダブルクリック
2 値を入力

COLUMN

オブジェクト名

Visual Basic では、フォームとコントロールのオブジェクト名を自由に変更できます。機能によって、プログラマー自身がわかりやすい名前に変更すれば、プログラムコードが読みやすくなるでしょう。

ただし、オブジェクト名を変更すると、そのオブジェクトのイベントプロシージャの名前も変わってしまいます。もし、すでにイベントプロシージャが存在している、フォームやコントロールのオブジェクト名を変更してしまったら、そのオブジェクトに対応するイベントプロシージャ名も同様に変更してください。そうしないと、イベントプロシージャが正しく呼び出されなくなります。

20 ツールボックスの［CommandButton］ をクリックし、フォーム上にコマンドボタンを配置する

Form1
登録済みアプリケーション
List1
追加　削除
Run　Command1

1 ここをクリックして…
2 そのままここまでドラッグ

21 プロパティウィンドウで［Caption］プロパティの値を「終了」に変更する

ﾌﾟﾛﾊﾟﾃｨ - Command1
Command1 CommandButton
全体 | 項目別

Appearance	1 - 3D
BackColor	&H8000000F&
Cancel	False
Caption	終了
CausesValidation	True
Default	False

1 ここをダブルクリック
2 値を入力

22 ［(オブジェクト名)］プロパティの値を「Exit」に変更する

ﾌﾟﾛﾊﾟﾃｨ - Exit
Exit CommandButton
全体 | 項目別

(ｵﾌﾞｼﾞｪｸﾄ名)	Exit
Appearance	1 - 3D
BackColor	&H8000000F&
Cancel	False
Caption	終了
CausesValidation	True

1 ここをダブルクリック
2 値を入力

コマンドボタンを選択

この操作を行う前に、作成したコマンドボタンコントロールが選択されていることを確認してください。

23 [ToolTipText] プロパティに「プログラムの終了」と入力する

Exit CommandButton
全体 | 項目別

TabStop	True
Tag	
ToolTipText	プログラムの終了
Top	3000

1 ここをダブルクリック
2 値を入力

24 フォームを選択し、[(オブジェクト名)] プロパティの値を「main」に変更する

フォームの選択

フォームを選択するには、フォーム上のコントロール以外の部分をクリックします。

main Form
全体 | 項目別

(オブジェクト名)	main
Appearance	1 - 3D
AutoRedraw	False
BackColor	&H8000000F&

1 ここをダブルクリック
2 値を入力

25 プロパティウィンドウで [Caption] プロパティの値を「プログラムランチャー」に変更する

main Form
全体 | 項目別

AutoRedraw	False
BackColor	&H8000000F&
BorderStyle	2 - 可変
Caption	プログラムランチャー

1 ここをダブルクリック
2 値を入力

26 完成したレイアウト

プログラムランチャー
登録済みアプリケーション
List1
追加
削除
Run
終了

コードの記述

main:(General)-(Declarations)

Project1 - main (コード)
(General) (Declarations)

```
Dim Exec(4) As String
Dim i As Integer
Dim AppCount As Integer
Dim AppIndex As Integer
```

main:AppAdd-Click

Project1 - main (コード)
AppAdd Click

```
Private Sub AppAdd_Click()
    Setup.Show 1
End Sub
```

main:AppDel-Click

Project1 - main (コード)
AppDel Click

```
Private Sub AppDel_Click()
    Dim LoopCount As Integer

    LoopCount = 1

    AppCount = GetSetting(appname:="QRun", section:="SetApp", _
                          Key:="AppCount", Default:="0")
    AppIndex = List1.ListIndex
    For i = 1 To AppCount
        DeleteSetting "QRun", "SetApp", i
    Next

    List1.RemoveItem AppIndex
    AppCount = AppCount - 1
    SaveSetting appname:="QRun", section:="SetApp", _
                Key:="AppCount", setting:=AppCount

    For LoopCount = 1 To AppCount
        SaveSetting appname:="QRun", _
                    section:="SetApp", Key:=LoopCount, _
                    setting:=List1.List(LoopCount - 1)
    Next
```

```
    If AppCount = 0 Then
        MsgBox "起動するアプリケーションを登録してください。"
        Setup.Show 1
    End If
    AppDel.Enabled = False
End Sub
```

main:Exit-Click

Project1 - main (ｺｰﾄﾞ)

Exit | Click

```
Private Sub Exit_Click()
    End
End Sub
```

main:Form-Load

Project1 - main (ｺｰﾄﾞ)

Form | Load

```
Private Sub Form_Load()
    AppCount = GetSetting(appname:="QRun", section:="SetApp", _
                         Key:="AppCount", Default:="0")
    If AppCount = 0 Then
        MsgBox "起動するアプリケーションを登録してください。"
        Setup.Show 1
    Else
        For i = 1 To AppCount
            Exec(i) = GetSetting(appname:="QRun", _
                      section:="SetApp", Key:=i, Default:="")
            List1.AddItem Exec(i)
        Next
    End If
        AppDel.Enabled = False
End Sub
```

main:List1-Click

Project1 - main (ｺｰﾄﾞ)

List1 | Click

```
Private Sub List1_Click()
    AppDel.Enabled = True
End Sub
```

main:Run-Click

Project1 - main (コード)

Run　Click

```
Private Sub Run_Click()
    Dim Ret As Variant

    AppCount = GetSetting(appname:="QRun", section:="SetApp", _
                          Key:="AppCount", Default:="0")

    For i = 1 To AppCount
        Exec(i) = GetSetting(appname:="QRun", section:="SetApp", _
                             Key:=i, Default:="")
        Ret = Shell(Exec(i), vbNormalFocus)
    Next
    main.WindowState = 1
End Sub
```

ワンポイントアドバイス

長いステートメントは、行継続文字「 _」(スペースとアンダースコアの組み合わせ)を使って、複数の行に分割入力することができます。

この文字を使えば、コードを画面に表示する横幅をせばめることができ、読みやすくなります。メッセージボックスに表示する文字が多いときなどに使用すると便利です。

メッセージなどの "(半角二重引用符)を途中で分割するには、二重引用符を閉じた後、次の行で「&」記号を追加、また引用符から続きを入力します。

行継続文字(_)は、最大1行に25個まで使用することができます。

COLUMN ……

レジストリ

「レジストリ」というのは、Windowsがシステム設定やアプリケーション情報を記録しておくための、データベース機能のことです。例えば、アプリケーションをインストールしたフォルダ名や、ユーザー登録してある名前などは、このレジストリに保管され、アプリケーションで使われています。実際にはWindowsフォルダの中にある「System.dat」と「User.dat」という2つのファイルがレジストリの正体です。

Windows 3.1の時は、「System.ini」「Win.ini」などの設定ファイルを、アプリケーション自身が個別に持っていました。そのため、Windowsのディレクトリにはこの「ini」ファイルがあふれてたのですが、Windows 95からは、このレジストリ方式を採用し、一括管理するようにしたため、アプリケーションが個別に設定ファイルを持つ必要がなくなりました。

解 説

1 このメインフォームの動作を説明しておきます。

①フォームが起動すると、Windowsのレジストリから、起動させたいアプリケーションのフルパスを取得し、リストボックスに登録します。
②登録するアプリケーションがない場合は、セットアップダイアログボックスを表示します。
③[Run] ボタンを押すと、登録したアプリケーションを起動します。起動後、プログラムは最小化されます。
④アプリケーションの登録は [追加] ボタンを押すと表示される [セットアップ] ダイアログボックスで行います。登録したアプリケーションは、Windowsのレジストリに保存されます。
⑤セットアップダイアログボックスでの処理は、次のレッスンで別に作成しますから、[追加] ボタンはダイアログボックスを表示する処理しか行いません。
⑥リストボックスに登録されたアプリケーションをクリックすると、[削除] ボタンが有効になり、リストボックスから削除することができます。同時に、Windowsのレジストリからも情報を削除し、レジストリを設定し直します。

アプリケーションを登録した状態

[セットアップ] ダイアログボックスを呼び出した状態

2　最初に、モジュールレベルの変数を宣言します。

```
Dim Exec(4) As String
Dim i As Integer
Dim AppCount As Integer
Dim AppIndex As Integer
```

「i」はループのカウンタ用の変数です。「AppCount」はWindowsのレジストリに保存してある、登録アプリケーションの数を保存する変数、「AppIndex」はリストボックスのリスト項目番号を格納するのに使用します。

ここでポイントになるのは、変数「Exec(4)」です。この変数だけ「(4)」が付いているのは、配列用変数として宣言しているからです。

「配列」というのは、同じ種類のデータをいくつもまとめたものをいいます。そして、この配列データを変数に格納する場合は、変数を配列で宣言しておきます。

3　変数「Exec(4)」は、登録するアプリケーション名をフルパスで格納します。4つまでのアプリケーションを格納しますから、変数を4つ用意しても構わないのですが、配列型の変数を使用すると、変数は1つで済みます。変数を配列で宣言する場合は、変数名の後に () 付きの数字を付け加えます。この数字は「要素数」や「インデックス」と呼ばれる数字で、全部でいくつの配列にするのかを指定します。「Exec(4)」と宣言すると、Exec(0)～Exec(4)までの、全部で5つのデータが格納できる変数を用意したことになります。

配列変数にデータを格納する場合は、何番目の配列に格納するのか、を指定します。配列の最初に格納する場合は、

```
Exec(0) = "Hello!"
```

となります。3番目の配列に格納するには、

```
Exec(2) = "Welcome"
```

という記述になります。配列番号は、必ず「0」から始まります。ですから、「Exec(4)」と宣言した配列変数は、「Exec(0)」から「Exec(4)」までの5つの配列になります。

配列の要素数が数字で指定できるということは、配列番号に変数を指定できるということです。さらに、変数で配列番号を指定できるということは、ループ処理などと組み合わせて、変数への値の代入を、式を使って自動化できるということにもなります。

配列名

Exec()

要素番号

0	
1	
2	
3	

「Hello!」を取り出すには「Exec(0)」を指定します。

「Welcome」を取り出すには「Exec(2)」を指定します。

4 変数宣言をしたら、フォームの初期化処理を組み立てます。

最初に、Windowsのレジストリを読み込み、アプリケーションがいくつ登録されているのかを調べ、変数「AppCount」に格納します。

```
AppCount = GetSetting(appname:="QRun", section:="SetApp", _
                      Key:="AppCount", Default:="0")
```

レジストリからデータを読み出すには、「GetSetting」関数を使います。

Windowsのレジストリは、全部で4つの項目で構成されています。

【GetSetting 関数の書式】

```
GetSetting(appname, section, Key[, Default])
```

appname	レジストリエントリの最初の名前です。主にアプリケーションの名前が使われますが、他の名前を自由に付けてもかまいません。
section	アプリケーション名のレジストリエントリには、複数の設定領域（レジストリエディタではキーを保存するフォルダとして表示される）を持てます。ここでは、その中の1つを指定します。
Key	レジストリキー（レジストリに設定する値）の名前を指定します。
Default	キー設定に数値が指定されていない場合（つまりレジストリに情報の登録が無い場合）に返す値を含む式を指定します。省略すると長さ0の文字列（""）となります。

この例題プログラムでは、「QRun − SetApp」にある「AppCount」というキー（次ページの上図参照）に、登録するアプリケーションの値を設定しています。これらの名前は、レジストリへの書き込み設定時に、プログラマが自由に付けることができます。レジストリにアクセスして値を取り出すには、GetSetting関数に、これらのレジストリエントリの名前とキーを指定して行います。代入記号は「=」ではなく「:=」を使います。

レジストリの構成は、Windowsがインストールされているフォルダにある、「regedit.exe」を起動して調べることができます。

レジストリテーブルに設定されたランチャープログラムの設定

COLUMN ……

レジストリエディタ

レジストリを操作するには、Visual Basicのようにアプリケーション側でレジストリデータの読み書きをする機能を組み込むか、専用のプログラム「regedit.exe」を使用します。このレジストリ操作プログラム「regedit.exe」を、「レジストリエディタ」と呼びます。作成したランチャープログラムのレジストリも、このレジストリエディタで設定を変更できます。

しかし、レジストリには、自分のプログラムで使う以外に、Windowsの動作を設定するための重要な情報も格納されているので、むやみに設定値を変更しないようにしてください。失敗するとWindows自体を破壊する恐れがあります。

5 アプリケーションが登録されていなければ、変数AppCountの値は「0」なので、メッセージボックスを表示し、セットアップダイアログボックスを表示します。

```
If AppCount = 0 Then
    MsgBox "起動するアプリケーションを登録してください。"
    Setup.Show 1
```

アプリケーションが登録されていれば、レジストリから順番に登録されているアプリケーション名を、配列変数「Exec(i)」に格納します。このとき、配列変数「Exec(i)」の配列番号は変数「i」を使用し、For...Nextループで配列にアプリケーション名を格納しています。

ループは、変数「AppCount」を使って、レジストリに登録してあるアプリケーションの数だけ回るようにしています（i = 1のときからAppCountの値まで）。

また、各アプリケーション名は、レジストリのキーを「1」「2」と数字で設定しているので、GetSetting関数の引数に変数「i」をそのまま使うことができます。

レジストリから読み込んだアプリケーション名は、AddItemメソッドを使って、リストボックスコントロールに追加していきます。

```
Else
    For i = 1 To AppCount
        Exec(i) = GetSetting(appname:="QRun", _
                             section:="SetApp", Key:=i, _
                             Default:="")
        List1.AddItem Exec(i)
    Next
End If
```

最後に、「削除」ボタンを動作できないようにしています。これは、リストボックスの項目が選択されていないのにボタンが押されると、プログラムがエラーを起こしてしまいますので、これを防止するためです。

```
AppDel.Enabled = False
```

Form_Loadの処理の流れ（要約）

6 ［追加］ボタンが押されると、セットアップダイアログボックスが表示されます。実際のアプリケーションの登録は、このセットアップダイアログボックスで行います（これは、次のレッスンで解説します）。

ここでは、Showメソッドを使って、ダイアログボックス用のフォームを表示するだけになります。

```
Private Sub AppAdd_Click()
    Setup.Show 1
End Sub
```

7

［削除］ボタンの処理は、やや複雑です。ボタンが押されると、最初にレジストリの登録をすべて消します。そして、リストボックスで選択された項目を、リストボックスから削除します。なぜ、最初にレジストリの登録をすべて抹消するかというと、リストボックスは、［Sorted］プロパティを「True」に設定してありますから、項目が削除されると自動的にリストが並べ替えられてしまいます。

そのため、レジストリに登録されている順番とリストボックスの順番が食い違ってしまいます。そこで、一旦レジストリの登録をすべて消して、リストに残された項目を再度そのままレジストリに登録し直します。

では、順番に説明していきましょう。

8

まず、プロシージャの先頭で、For...Nextループのための変数「LoopCount」を宣言します。このプロシージャでは、2種類のFor...Nextループを使用しますので、その1つのループカウンタに使用します。

宣言したら、「1」を代入して初期化しておきます。

```
Private Sub AppDel_Click()
    Dim LoopCount As Integer

    LoopCount = 1
```

9

続いて、GetSetting関数を用いて、レジストリに保存されているアプリケーションの数「AppCount」を読み出し、変数AppCountに格納します。これは、登録アプリケーションが削除されたとき、残りの登録数を計算するために使用します。

また、現在リストボックスコントロールで選択されている項目を把握するために、［ListIndex］プロパティの値を変数「AppIndex」に格納します。

この［ListIndex］プロパティは、現在選択されているリスト項目のインデックス番号を格納しているプロパティで、リストボックスから項目を削除するときに使用します。

```
AppCount = GetSetting(appname:="QRun", section:="SetApp", _
                      Key:="AppCount", Default:="0")
AppIndex = List1.ListIndex
```

10

For...Nextループを用いて、レジストリテーブルから、登録してあるアプリケーションをすべて削除します。変数AppCountの値を使って、登録数分の回数のループを回します。

レジストリからキーと値を削除するには、DeleteSettingステートメントを使用します。GetSettingと同様にパラメータに、レジストリのappname、section、keyを指定します。

```
For i = 1 To AppCount
    DeleteSetting "QRun", "SetApp", i
Next
```

【DeleteSettingステートメントの書式】

```
DeleteSetting appname, section[, key]
```

appname	削除するレジストリのアプリケーション名を指定します。
section	削除するキーが設定されたセクション名を指定します。
key	削除するキー名を含む文字列を指定します。省略可能。

11 レジストリを削除したら、リストボックスから項目名を削除します。これは、RemoveItemメソッドを使用します。引数には、リスト項目の番号を指定します。

```
List1.RemoveItem AppIndex
```

リストボックスコントロールの［Sorted］プロパティを「True」に設定してあるので、削除された時点でリスト項目は昇順に並べ替えられます。そして、登録アプリケーションが1つ減りますから、変数AppCountの値を1つ減らしておきます。

```
AppCount = AppCount - 1
```

プログラムランチャー
登録済みアプリケーション
c:¥WINNT¥SYSTEM32¥CALC.EXE
c:¥WINNT¥SYSTEM32¥cdplayer.exe
c:¥WINNT¥SYSTEM32¥RASPHONE.EXE
追加
削除
Run
終了

AppCount = 3

1 項目を削除すると…

プログラムランチャー
登録済みアプリケーション
c:¥WINNT¥SYSTEM32¥CALC.EXE
c:¥WINNT¥SYSTEM32¥cdplayer.exe
追加
削除
Run
終了

2 残りが昇順に並べ替えられる

AppCount = 3 − 1= 2

COLUMN ・・・・・

リストボックスの項目の削除

リストボックスの項目を削除する場合、RemoveItemメソッドを使いますが、削除する項目の指定には、リスト項目の番号を使います。AddItemメソッドは、追加する項目名を使いましたが、RemoveItemメソッドでは項目番号で指定するので、間違えないでください。

12

リストボックスの処理が終わったら、リスト項目をすべてレジストリへ書き出し、変更内容をレジストリに再保存します。これは、登録アプリケーションが削除されれば、ここでレジストリに設定したアプリケーションの数も変更するためです。

レジストリへの書き込みは、「SaveSetting」ステートメントを使用します。パラメータとして、レジストリエントリの名前とセクション、キーとその値を順番に指定します。ここでは、キーに格納する値（Setting）は、登録するアプリケーションの数を設定します。アプリケーションの数は、「QRun－SetApp－AppCount」というレジストリエントリとキーを設定しているので、リストボックスに登録したアプリケーションの数をキー「AppCount」に値として保存します。

```
SaveSetting appname:="QRun", section:="SetApp", _
        Key:="AppCount", setting:=AppCount
```

13

SaveSettingステートメントは、Windows 95/98（またはWindows NT 4.0）のレジストリデータベースに、アプリケーション固有の情報を保存するステートメントです。ステートメントのパラメータに、レジストリの名称、その中のセクション、そしてキーと値を自由に設定して使うことができます。

【SaveSettingステートメントの書式】

```
SaveSetting appname, section, key, setting
```

appname	設定するアプリケーションの名前を文字列で指定します。
section	保存するキーを設定するセクションの名前を文字列で指定します。
key	値を格納するキーを文字列で設定します。
setting	キーに格納する値、または値を含む式を指定します。

通常、アプリケーション1つに対して1つの「appname」を設定します。そして、この中には複数の「section」を設定できます。このプログラムでは、アプリケーションの登録数と、登録アプリケーションのフルパス名の、2種類のsectionを設定しています。sectionの中には、さらにいくつものキーを設定し、1つのキーに対して1つの値を持つことができます。

レジストリは、Windowsのシステムやコンピュータにインストールされた各アプリケーションが自分の領域を設定するために使用しています。Visual Basicのユーザープログラムがレジストリを使える領域も決まっており、筆者のWindows 98では「HKEY_CURRENT_USER¥Software¥VB and VBA Program Settings」の中に設定されています。

ただし、このレジストリはあくまでも1つの例であり、使用するコンピュータシステムやWindows NT 4.0など他のOSでは少し違ってくる場合もあります。Windowsのフォルダに、「regedit.exe」というレジストリを読み書きできるプログラムがあるので、これを使って自分のコンピュータのレジストリを調べてください。

[編集] メニューから [検索] を選択

「QRun」を検索する

Windows 98のレジストリテーブル

ここに保存されている

14

アプリケーションの数を登録したら、続いて実際の登録アプリケーションをフルパスでレジストリに保存します。

リストボックスの［List］プロパティには、リストボックスに登録したアプリケーションがフルパスで記述されていますから（List1.List(LoopCount-1)）、For...Nextループを使ってリストボックスのすべてのアプリケーションをレジストリに1つずつ書き出していきます。

これで、リストボックスに登録されているアプリケーションの情報は、すべて更新されてレジストリに保存されます。

```
For LoopCount = 1 To AppCount
    SaveSetting appname:="QRun", section:="SetApp", _
                Key:=LoopCount, _
                setting:=List1.List(LoopCount - 1)
Next
```

15 もし、リストボックスの項目がすべて削除されると、再度アプリケーションを登録するようにメッセージボックスを表示します。メッセージボックスの［OK］ボタンが押されると、セットアップ用のダイアログボックスを表示します。

そして、［削除］ボタンのプロシージャの最後で、［削除］ボタンの無効処理をします。これは、リストボックスの項目が選択されていないのに、［削除］ボタンが動作するのを防止するためです。

```
If AppCount = 0 Then
    MsgBox "起動するアプリケーションを登録してください。"
    Setup.Show 1
End If
AppDel.Enabled = False
```

COLUMN ・・・・・

4つ以上登録できないようにするには

ここで解説したコードを実行すると、4つ以上アプリケーションを登録できますが、いったんプログラムを終了して、再度起動するとエラーになってしまいます。これは有効な配列変数Exec()が用意されていないことが原因です。

ユーザーが4つ以上のアプリケーションを登録しようとしたら、メッセージボックスを表示して登録させないようにするには、［追加］ボタンにコードを記述するといいでしょう。「Private Sub AppAdd_Click」プロシージャのコードを以下のように変更してください（この処理はサンプルプログラムには追加されています）。

```
If List1.ListCount < 4 then
    Setup.Show 1
Else
   MsgBox "4つ以上は登録できません"
End If
```

AppDel_Clickの処理の流れ（要約）

LoopCountに1を代入

レジストリに保存されているアプリケーション数をGetSetting関数で取得

リストボックスの項目数をAppIndexに代入

i=1からAppCount値まで繰り返し

DeleteSetting関数でレジストリ情報削除

For...Next文

次の繰り返し

繰り返し終了

リストボックスから選択された項目名を削除

AppCountから1減らす

減少したアプリケーションの数を再登録

LoopCount=1からAppCount値まで繰り返し

リストボックスのすべてのアプリケーション名をレジストリに書き出し保存

繰り返し終了

次の繰り返し

登録が0の時は登録処理を実行

削除ボタンを無効

16

今度は、登録したアプリケーションプログラムを実行させる［Run］ボタンの処理を作成します。

まず、レジストリから、登録されているアプリケーション名をフルパスで取得します。フォームのLoad()イベントプロシージャでも、同じようにレジストリから保存内容を取得していますが、これはリストボックスの初期状態の作成に使用しているためです。その後、ユーザーによってアプリケーションの追加や削除が行われている可能性がありますから、［Run］ボタンが押された時点で、再度レジストリを読み出して、最新の情報を取得します。

読み出したアプリケーションのパスと名前は、配列で準備した変数「Exec」に、For...Nextループを使って、順番に格納します。ループの回数は、レジストリにあるキー「AppCount」に保存されている、アプリケーションの登録数を使います。

```
Private Sub Run_Click()
    Dim Ret As Variant

    AppCount = GetSetting(appname:="QRun", _
                          section:="SetApp", _
                          Key:="AppCount", Default:="0")

    For i = 1 To AppCount
        Exec(i) = GetSetting(appname:="QRun", _
                             section:="SetApp",
                             Key:=i, Default:="")
        Ret = Shell(Exec(i), vbNormalFocus)
    Next
    main.WindowState = 1
End Sub
```

17

他のプログラムを呼び出して実行するのは、Shell関数で行います。この関数は、引数に実行したいプログラム名と表示形態を指定します。

```
Ret = Shell (Exec (i) , vbNormalFocus)
```

【Shell 関数の書式】

```
Shell (pathname[, window])
```

pathname	実行するプログラム名のフルパス名を指定します。
window	表示するプログラムのウィンドウの形態を定数で指定します。省略可能です。

プログラム名はフルパスで記述し、表示形式は定数（以下の表参照）を指定します。Shell関数は、指定したプログラムを実行すると、プログラムのタスクIDというもの（Windowsがシステム管理で使用しているデータ）を返してきますので、これをバリアント（Variant）型の変数で受け取ります。変数Retは、戻り値を受け取るだけで、特に何も使いません。

＜Shell関数で使えるwindow定数値一覧＞

記述指定（文字定数）	数値指定	表示されるボタンの内容
vbHide	0	非表示ウィンドウ
vbNormalFocus	1	通常表示に復元されるウィンドウ
vbMinimizedFocus	2	最小化表示（省略時設定）
vbMaximizedFocus	3	最大化表示
vbNormalNoFocus	4	通常表示に復元されるウィンドウをバックグラウンド表示
vbMinimizedNoFocus	6	バックグラウンドに最小化表示

18

このプロシージャの最後に、フォームの［WindowState］プロパティを使って、フォームをアイコン化します。［WindowState］プロパティは、フォームの表示形態を設定します。このプロパティは、Visual Basicのプロパティウィンドウを使って設定できますが、このようにコードからでも設定できます。プログラムの動作状態によって、フォームをアイコン化したり、最大化することができます。ここでは、アプリケーション起動後に、プログラムそのものをタスクバーに格納し、アイコンで表示（vbMinimized）するように指定しています。

```
main.WindowState = 1
```

＜WindowState定数値一覧＞

記述指定（文字定数）	数値指定	表示されるボタンの内容
vbNormal	0	ノーマル表示（既定値）
vbMinimized	1	最小化表示
vbMaximized	2	最大化表示

19

リストボックスでは、項目が選択されるとClickイベントが発生しますから、このときはじめて［削除］ボタンを有効にする処理を行います。

そして、最後に［終了］ボタンが押されたときの処理を記述して、完成です。

```
Private Sub List1_Click()
    AppDel.Enabled = True
End Sub

Private Sub Exit_Click()
    End
End Sub
```

プログラムランチャー

登録済みアプリケーション

c:¥WINNT¥SYSTEM32¥CALC.EXE
c:¥WINNT¥SYSTEM32¥cdplayer.exe

追加　削除

AppDel.Enabled = False

項目が選択されないと［削除］ボタンは無効

プログラムランチャー

登録済みアプリケーション

c:¥WINNT¥SYSTEM32¥CALC.EXE
c:¥WINNT¥SYSTEM32¥cdplayer.exe

追加　削除

AppDel.Enabled = True

項目が選択されてはじめて［削除］ボタンが有効になる

まとめ

これまでは、「*.ini」などのプロファイル・ファイルにその設定を保存して、Windowsのディレクトリに格納していましたが、Windows 95/98ではレジストリを使うようになっています。レジストリは、自分のプログラムの設定を保存しておくのに大変便利なシステムです。

保存するにはSaveSettingステートメント、読み出すにはGetSetting関数を使用するのが特徴です。

また、レジストリの読み書きステートメントは、自分のプログラムで使う領域しかアクセスできません。他のアプリケーションの情報などは取得できないので注意してください。

Lesson 25(2)

第６日
２時限目

ソフトウェアランチャーの作成

Lesson25.vbp
VB6CD
Lesson25

◎Learning Edition

セットアップモジュールの作成

ファイル関係のリストボックスの使い方を復習します。また、Windowsのレジストリの読み書きを復習します。

今回作成するプログラム

実行プログラムの登録
c: [IDE-BOOT]
c:¥
WINNT
SYSTEM32
CACHE
COLOR
CONFIG
DHCP
DRIVERS
INETSRV
LogFiles
Macromed
OS2
NET1.EXE
NETDDE.EXE
NETSTAT.EXE
nlsfunc.exe
notepad.exe
NSLOOKUP.EXE
NTBACKUP.EXE
NTOSKRNL.EXE
NTVDM.EXE
ODBCAD32.EXE
ODBCCONF.EXE
キャンセル
追加

1 プログラムを選択して…
2 ［追加］をクリックすると…

プログラムランチャー
登録済みアプリケーション
c:¥WINNT¥SYSTEM32¥CALC.EXE
c:¥WINNT¥SYSTEM32¥cdplayer.exe
c:¥WINNT¥SYSTEM32¥notepad.exe
追加
削除
Run
終了

3 プログラムが登録される…
4 ［Run］をクリックするとプログラムが起動する

POINT

ここでは、既に説明したドライブ・ディレクトリ・ファイルリストボックスの３つのリストボックスを使って、選択したアプリケーションをWindowsのレジストリに書き込みます。

また、メインフォームのリストボックスへの追加も行います。

アプリケーションの追加登録処置は、単独のユーザー定義プロシージャを用意し、それをコントロールのイベントプロシージャに配置する方法を使います。

コントロールとプロパティの設定

1 前回までのプロジェクトに［フォームモジュール］を追加する

Project1 - Microsoft Visual Basic [デザイン]
ファイル(F) 編集(E) 表示(V) プロジェクト(P) 書式(O)
フォーム モジュール(F)
MDI フォーム モジュール(I)
標準モジュール(M)
クラス モジュール(C)
ユーザー コントロール(U)
プロパティ ページ(R)

1 ここをクリック

2 フォームモジュールを選択し、［開く］ボタンをクリックする

フォーム モジュールの追加
新規作成 | 既存のファイル
フォーム モジュール　データ フォーム ウィザード　バージョン情報　Web ブラウザ　ダイアログ　ログオン ダイアログ
見出し画面　ワンポイント　ODBC ログイン　オプション ダイアログ
開く(O)　キャンセル　ヘルプ(H)
次回からこのダイアログ ボックスを表示しない(D)

1 ［フォームモジュール］を選択
2 このボタンをクリック

3 プロパティウィンドウで［Caption］プロパティの値を「実行プログラムの登録」に変更する

プロパティ - Form1
Form1 Form
全体 | 項目別

AutoRedraw	False
BackColor	&H8000000F&
BorderStyle	2 - 可変
Caption	実行プログラムの登録
ClipControls	True
ControlBox	True
DrawMode	13 - Copy Pen

1 ここをダブルクリック
2 値を入力

今回作成するプログラム

アプリケーションを登録するための、セットアップ用モジュールを作成します。

セットアップは、専用のフォームを用意し、そこにドライブ・ディレクトリ・ファイルリストボックスを使って、登録するアプリケーションを選択できるようにします。

フォームモジュールの追加画面

この追加画面はVisual BasicのEditionによって表示が異なります。

ドライブリストボックス

ドライブを選択するときに使用するコントロールです。ユーザーがドライブを選択するとDriveプロパティが変更されます。

ディレクトリリストボックス

フォルダ（ディレクトリ）を選択するときに使用するコントロールです。ユーザーがフォルダをダブルクリックすると、Pathプロパティが変更されます。

ファイルリストボックス

フォルダ内のファイルをリスト表示するコントロールです。ユーザーがファイルを選択するとPathプロパティが変更されます。
パターン（Pattern）というプロパティを設定して特定の拡張子のみをフィルタリング表示させることも可能です。

4 ツールボックスの［DriveListBox］をクリックし、フォーム上にドライブリストボックスを配置する

1 ここをクリックして…

2 そのままここまでドラッグ

5 ツールボックスの［DriListBox］をクリックし、フォーム上にディレクトリリストボックスを配置する

1 ここをクリックして…

2 そのままここまでドラッグ

6 ツールボックスの［FileListBox］をクリックし、フォーム上にファイルリストボックスを配置する

1 ここをクリックして…

2 そのままここまでドラッグ

7 プロパティウィンドウで［Pattern］プロパティの値を「*.exe」に変更する

プロパティ - File1
File1 FileListBox
全体 | 項目別

Normal	True
OLEDragMode	0 - 手動
OLEDropMode	0 - なし
Pattern	*.exe
ReadOnly	True

1 ここをダブルクリック
2 「*.exe」を入力

8 ツールボックスの［CommandButton］をクリックし、フォーム上にコマンドボタンを配置する

実行プログラムの登録
i: [SDK & DEV60]
I:¥
Program Files
VB98
Template
Tsql
Wizards
C2.EXE
CVPACK.EXE
LINK.EXE
VB6.EXE
VISDATA.EXE
Command1

1 ここをクリックして…
2 そのままここまでドラッグ

9 プロパティウィンドウで［Caption］プロパティの値を「キャンセル」に変更する

プロパティ - Command1
Command1 CommandButton
全体 | 項目別

(オブジェクト名)	Command1
Appearance	1 - 3D
BackColor	&H8000000F&
Cancel	False
Caption	キャンセル
CausesValidation	True

1 ここをダブルクリック
2 「キャンセル」を入力

ファイルリストボックスを選択

この操作を行う前に、作成したファイルリストボックスコントロールが選択されていることを確認してください。

コマンドボタンを選択

この操作を行う前に、作成したコマンドボタンコントロールが選択されていることを確認してください。

10 [(オブジェクト名)] プロパティの値を「cxsetup」に変更する

11 ツールボックスの [CommandButton] をクリックし、コマンドボタンをもう1つ配置する

12 プロパティウィンドウで [Caption] プロパティの値を「追加」に変更する

13 [(オブジェクト名)] プロパティの値を「AddProgram」に変更する

14 フォームを選択し、[(オブジェクト名)] プロパティの値を「Setup」に変更する

プロパティ - Setup
Setup Form
全体 | 項目別

(オブジェクト名)	Setup
Appearance	1 - 3D
AutoRedraw	False
BackColor	&H8000000F&

1 ここをダブルクリック
2 値を入力

15 「filemanager.frm」という名前でフォームを保存する

名前を付けてファイルの保存
保存する場所(I): Lesson25
main.frm
ファイル名(N): filemanager.frm
ファイルの種類(T): フォーム モジュール (*.frm)
保存(S)　キャンセル　ヘルプ(H)

1 名前を入力
2 ここをクリック

16 完成したプロジェクトファイル

プロジェクト - Project1
Project1 (Lesson25.vbp)
フォーム
main (main.frm)
Setup (filemanager.frm)

フォームの選択

フォームを選択するには、フォーム上のコントロール以外の部分をクリックします。

プロジェクトフォームの名前の確認

プロジェクトフォームの名前は、プロジェクトエクスプローラで確認することができます。

コードの記述

Setup:(General)-(Declarations)

Project1 - Setup (コード)

(General) | (Declarations)

```
Dim i As Integer
```

Setup:AddProgram-Click

Project1 - Setup (コード)

AddProgram | Click

```
Private Sub AddProgram_Click()
    Call SetupApp
End Sub
```

Setup:cxsetup-Click

Project1 - Setup (コード)

cxsetup | Click

```
Private Sub cxsetup_Click()
    Setup.Hide
End Sub
```

Setup:Dir1-Change

Project1 - Setup (コード)

Dir1 | Change

```
Private Sub Dir1_Change()
    File1.Path = Dir1.Path
End Sub
```

Setup:Drive1-Change

Project1 - Setup (コード)

Drive1 | Change

```
Private Sub Drive1_Change()
    Dir1.Path = Drive1.Drive
End Sub
```

Setup:File1-DblClick

Project1 - Setup (コード)　File1　DblClick

```
Private Sub File1_DblClick()
    Call SetupApp
End Sub
```

Setup:Form-Load

Project1 - Setup (コード)　Form　Load

```
Private Sub Form_Load()
    i = GetSetting(appname:="QRun", section:="SetApp", _
                   Key:="AppCount", Default:="0")
End Sub
```

Setup:(General)-SetupApp

Project1 - Setup (コード)　(General)　SetupApp

```
Private Sub SetupApp()
    Dim Ret As String
    Dim ExecFile As String

    ExecFile = Dir1.Path & "¥" & File1.filename
    Ret = MsgBox("選択したアプリケーションを登録しますか？", _
                 vbOKCancel, "アプリケーションの登録")
    If Ret = vbOK Then
        i = GetSetting(appname:="QRun", section:="SetApp", _
                       Key:="AppCount", Default:="0")
        i = i + 1
        SaveSetting appname:="QRun", section:="SetApp", _
                    Key:=i, setting:=ExecFile
        main.List1.AddItem ExecFile
    End If

    SaveSetting appname:="QRun", section:="SetApp", _
                Key:="AppCount", setting:=i

    Setup.Hide
End Sub
```

解　説

1　ドライブ、ディレクトリ、ファイルの各リストボックスは、すでに使用方法を説明したように（Lesson24参照）、［Drive］プロパティと［Path］プロパティを連動させて、ユーザーが選択したドライブとディレクトリの内容を表示できるようにします。

```
Private Sub Drive1_Change()
    Dir1.Path = Drive1.Drive
End Sub

Private Sub Dir1_Change()
    File1.Path = Dir1.Path
End Sub
```

2　ファイルリストボックスは、ユーザーがマウスでダブルクリックしたときに、イベント「DblClick」を発生します。

ユーザーがリストのファイル名をダブルクリックすると、そのままアプリケーションとして登録できるように、このイベントプロシージャでセットアップ操作のプロシージャ（SetupApp）を呼び出します。

```
Private Sub File1_DblClick()
    Call SetupApp
End Sub
```

3　［追加］ボタンの処理も、このセットアップ用のプロシージャ（SetupApp）を呼び出します。同じ作業を複数のプロシージャから呼び出す場合は、このように1つのプロシージャに処理をまとめて、それを呼び出すだけにしておくと、同じコードをいくつも書かなくてすみます。

```
Private Sub AddProgram_Click()
    Call SetupApp
End Sub
```

4　アプリケーションの追加登録があった場合は、何番目のアプリケーションになるのかを把握するために、レジストリに保存されている登録アプリケーションの数を、GetSetting関数を使って読み込んでおきます。

この処理は、フォームが表示されたときに実行されるように、フォームのLoadイベントプロシージャで行っておきます。

```
Private Sub Form_Load()
    i = GetSetting(appname:="QRun", section:="SetApp", _
                    Key:="AppCount", Default:="0")
End Sub
```

5 アプリケーションを追加登録する処理は、「SetupApp」というプロシージャを作成して、ここに記述します。

コントロールのイベントプロシージャではなく、独自のプロシージャを作成する場合は、キーワード「Private Sub」を付けてプロシージャを作成します。

作成したプロシージャは、コントロールのイベントプロシージャのように自動的に呼び出されるものではないため、呼び出す場所で「Call」ステートメントを使って、明示的に呼び出します。

6 プロシージャ「SetupApp」では、追加するアプリケーションのレジストリ保存とメインフォームのリストボックスへの組み込み処理を行います。

最初に、ディレクトリリストボックスの［Path］プロパティと、ファイルリストボックスの［filename］プロパティを使って、登録するアプリケーションの名前をフルパスで作成します。作成したファイル名は、変数「ExecFile」に格納します。

```
Private Sub SetupApp()
    Dim Ret As String
    Dim ExecFile As String

    ExecFile = Dir1.Path & "¥" & File1.filename
```

7 ファイル名が作成できたら、メッセージボックスでユーザーに再度登録するかどうかを尋ねます。［OK］ボタンが押されれば（Ret=vbOK)、ファイル名をレジストリに保存し、メインフォームのリストボックス（main.List1）にAddItemメソッドを使って組み込みます。このとき、フォームのオブジェクト名まできちんと指定してください。

```
Ret = MsgBox("選択したアプリケーションを登録しますか？", _
             vbOKCancel, "アプリケーションの登録")
If Ret = vbOK Then
    i = GetSetting(appname:="QRun", section:="SetApp", _
                   Key:="AppCount", Default:="0")
    i = i + 1
    SaveSetting appname:="QRun", section:="SetApp", _
                Key:=i, setting:=ExecFile
    main.List1.AddItem ExecFile
End If
```

8 アプリケーションの登録が終わったら、再度登録アプリケーションの数をレジストリに保存し、セットアップダイアログボックスを閉じます（Setup.Hide）。

```
SaveSetting appname:="QRun", section:="SetApp", _
            Key:="AppCount", setting:=i
Setup.Hide
```

9 セットアップダイアログボックスの［キャンセル］ボタンが押された場合は、何の処理もしませんから、そのままセットアップダイアログボックスを閉じて、プログラムの制御をメインフォームに戻します。

```
Private Sub cxsetup_Click()
    Setup.Hide
End Sub
```

10 以上で、プログラムが完成です。実行ボタンを押して、プログラムを動作させます。好きなアプリケーションを登録して、［Run］ボタンを押すと、登録したアプリケーションが次々と起動します。そして、プログラム自身は最小化されタスクバーに格納されます。

まとめ

レジストリデータベースを使ったアプリケーションをはじめて作成しました。このように、アプリケーションで常に使用するデータ（情報）は、レジストリを使うと有効です。

また、同じ処理を複数のコントロールやイベントで使う場合は、１つのプロシージャにまとめて、Callステートメントを使って、それを呼び出す形にします。こうしておけば、プロシージャを重複して記述することがなく、デバッグもしやすくなります。

このように、よく使う処理を汎用のプロシージャとして作成しておけば、他のプログラムにも使えるプロシージャになりますから、プログラム作成資源の再利用にもつながります。

ワンポイントアドバイス

このプログラムでは、登録できる数を４つに制限しましたが、これは登録するアプリケーションの数が、配列変数「Exec()」の配列数に依存するためです。

この配列数をもっと増やせば、その分多くのアプリケーションを登録できます。

練習問題

Q1　「Lesson25.vbp」に手を加えて、プログラム終了時にプログラムのバージョン番号を保存するようにしてください。

レジストリに設定されたバージョン番号

【ヒント】

レジストリテーブルは、これまで使ったAppNameを使います。
ここでは、以下のセクションとキーを使います。値は文字列にします。

appname:="QRun"
section:="Version"
Key:="Build"

また、プログラムのバージョン番号は、Appオブジェクトの［Major］［Minor］［Revisoin］プロパティを使います。

これらのプログラムのバージョン番号は、Visual Basicの［プロジェクト］メニューの［Project1のプロパティ］を選ぶと表示されるダイアログボックスの、［実行可能ファイルの作成］タブで設定できます。ここで設定されたバージョン番号が、Appオブジェクトのプロパティに格納される仕組みになっています。

このバージョン番号の使い方は、この後のレッスンでも紹介します。

バージョン番号設定ダイアログボックス

解答は巻末に→

COLUMN・・・・・

ドライブリストボックスでエラーがでるときは

このサンプルでは、ドライブリストボックスの選択で、使用されていない無効なドライブ（メディアが挿入されていないフロッピードライブやCD-ROMドライブ）を選択してしまうとエラーがでます。このエラーを回避するにはOn Errorステートメントを利用します。On Errorステートメントが実行されると、エラー処理モードが有効になり、エラーが発生すると「GoTo」で指定したラベルまで処理をジャンプさせます。

Visual Basicでは、エラーが発生するとErrオブジェクトが生成されます。このErrオブジェクトはエラーに関するいろいろな情報を保有しています。その中で、Errオブジェクトの［Err.Number］プロパティを使用すると、発生したエラーに対応するエラー番号を取得できます。このプロパティを使用すると、エラー番号を使って処理を分岐し、ユーザーへの注意を即すメッセージを表示したり、あるいはその処理を無効にする対策を作成できます。

また、Error関数は、エラー番号に対応したエラーメッセージを返す関数です。この関数を使用すれば、あらかじめエラー番号に対応したエラーメッセージを使用することができます。

このサンプルで発生する、「デバイスが準備されていない」ことを示すエラーの番号は「68」です（「付録　エラー番号一覧」を参照）。これを使ってエラーを処理します。ドライブリストボックスのChangeプロシージャを以下のように変更してください。この処理では、もしユーザーがフロッピーディスクの入っていないAドライブなどを選択すると、「デバイスが準備されていません。」というメッセージが表示されます。

エラー処理の実際についてはLesson32以降で詳しく説明しています。

```
Private Sub Drive1_Change()
    On Error GoTo FAIL
    Dir1.Path = Drive1.Drive
    Exit Sub

FAIL:
    MsgBox Error(Err.Number)
End Sub
```

Lesson25

デバイスが準備されていません。

OK

エラー番号に対応したエラーメッセージが表示される

Let's VBSchool!!

第７日 （Lesson26～Lesson31）
いろいろなテクニック

ここでは、Windowsの機能を活用するコントロールを中心に、いろいろなテクニックを紹介します。

●コモンダイアログボックスコントロール
●マルチメディアコントロール
●データベースコントロール
●タブ付きダイアログボックスコントロール
●フォームテンプレートの活用

今日のレッスンで解説する項目

❏ コモンダイアログボックスコントロールの設定方法
❏ ダイアログボックスの戻り値の使い方
❏ コモンダイアログボックスコントロールを活用したアプリケーションの作成
❏ マルチメディアコントロールのプロパティやメソッドの使い方
❏ コードからサウンドファイルを再生する方法
❏ データベースコントロールの初歩的な使い方
❏ 連結コントロールを操作してデータを表示する方法
❏ タブ付きダイアログボックスの使い方
❏ 各ページに配置したコントロールの操作方法とフォームからの呼び出し方法
❏ フォームテンプレート「見出し画面」の作成方法と表示方法

Lesson 26

第７日
１時限目

いろいろな
テクニック

Lesson26.vbp
VB6CD
Lesson26

◎Learning Edition

コモンダイアログボックス

コモンダイアログボックスコントロールの設定方法、ダイアログボックスの戻り値の使い方、コモンダイアログボックスコントロールを活用したアプリケーションの作成などを学習します。

● 今回作成するプログラム

Form1

ファイルを開く

フォントの設定

色の設定

このフォントはMS ゴシック

各ボタンをクリックすると、対応するダイアログボックスが表示される

[色の設定]

[フォントの指定]

[ファイルを開く]

POINT

　コモンダイアログボックスコントロールを組み込むと、Windowsが標準で装備している、[ファイルを開く] [名前を付けて保存] [色の設定] などのダイアログボックスを、そのまま使用することができます。ここでは、コモンダイアログボックスコントロールの使い方を理解するために、３つのダイアログボックスを表示するプログラムを作成します。

コントロールとプロパティの設定

※新しいプロジェクトとフォームを用意します。

1 ツールボックスをマウスの右ボタンでクリックし、メニューから［コンポーネント］を選ぶ

1 ここをクリック

2 ［コントロール］タブで［Microsoft Common Dialog Control 6.0］をチェックする

1 ここをチェック

2 ここをクリック

3 ツールボックスに［CommonDialog］ボタンが追加される

追加された［CommonDialog］ボタン

コモンダイアログボックスコントロール

コモンダイアログボックスコントロールは、Visual Basicの初期設定では、ツールボックスに組み込まれていません。使用する際に、ツールパレットにコントロールを組み込む必要があります。

配置は自由

タイマーコントロールと同様に、コモンダイアログボックスコントロールはフォーム上のどの位置に置いてもかまいません。

4 ツールボックスの［CommonDialog］をクリックし、フォーム上に配置する

Form1

1 ここをクリックして…

2 そのままここまでドラッグ

5 ツールボックスの［CommandButton］をクリックし、フォーム上にコマンドボタンを配置する

Form1

Command1

1 ここをクリックして…

2 そのままここまでドラッグ

コマンドボタンを選択

この操作を行う前に、作成したコマンドボタンコントロールが選択されていることを確認してください。

6 プロパティウィンドウで［Caption］プロパティの値を「ファイルを開く」に変更する

プロパティ - Command1

Command1 CommandButton

全体 | 項目別

(オブジェクト名)	Command1
Appearance	1 - 3D
BackColor	&H8000000F&
Cancel	False
Caption	ファイルを開く

1 ここをダブルクリック

2 値を入力

7 同様に、フォーム上にコマンドボタンを２つ配置し、［Caption］プロパティを次のように変更する

コマンドボタン	［Caption］プロパティ
Command2	フォントの設定
Command3	色の設定

8 ツールボックスの［Label］ボタン A をクリックし、フォーム上にラベルを配置する

1 ここをクリックして…

2 そのままここまでドラッグ

コードの記述

Form1:Command1-Click

Project1 - Form1 (コード)

Command1 / Click

```
Private Sub Command1_Click()
    Dim GetName As String

    With CommonDialog1
        .Filter = "すべてのファイル (*.*)|*.*"
        .ShowOpen
        GetName = .filename
    End With

    Label1.Caption = "選択したファイル名は" & GetName
End Sub
```

Form1:Command2-Click

Project1 - Form1 (コード)

Command2 / Click

```
Private Sub Command2_Click()
    With CommonDialog1
        .Flags = cdlCFBoth
        .ShowFont

         FName = .FontName
         FSize = .FontSize
    End With
    With Label1
        .Caption = "このフォントは" & FName
        .FontSize = FSize
    End With
End Sub
```

Form1:Command3-Click

Project1 - Form1 (コード)

Command3 / Click

```
Private Sub Command3_Click()
    CommonDialog1.ShowColor
    Label1.ForeColor = CommonDialog1.Color
End Sub
```

解 説

1 　コモンダイアログボックスコントロールは、どのダイアログボックスを使用するのかで、呼び出すメソッドが違います。

メソッド	表示されるダイアログボックス
ShowOpen	［ファイルを開く］ダイアログボックス
ShowSave	［ファイル名を付けて保存］ダイアログボックス
ShowColor	［色の設定］ダイアログボックス
ShowFont	［フォントの指定］ダイアログボックス
ShowPrinter	［印刷]］または［プリンタの設定］ダイアログボックス
ShowHelp	Windowsのヘルプエンジン

例えば、［ファイルを開く］ダイアログボックスは、次のメソッドを使用します。

```
CommonDialog1.ShowOpen
```

2 　コモンダイアログボックスが表示されて、ユーザーが何かを選択すると、選択内容は専用のプロパティに格納されます。そのプロパティの値を取得し、プログラムの処理に組み合わせていきます。このプログラムでは、［ファイルを開く］［フォントの指定］［色の設定］という3つのダイアログボックスを使ってみます。

3 　まず、［ファイルを開く］ダイアログボックスを使います。ダイアログボックスでファイルを選択すると、そのファイル名がラベルコントロールに出力されます。この一連の処理はすべて、［ファイルを開く］ボタンのClickイベントプロシージャに作成します。

プロシージャの先頭で、文字列型変数「GetName」を用意します。この変数に、［ファイルを開く］ダイアログボックスで選択されたファイル名を格納します。

```
Private Sub Command1_Click()
    Dim GetName As String
```

コモンダイアログボックスの処理（［ファイルを開く］の場合）

①ダイアログボックス表示 ← コマンドボタンをクリック
↓ ← ユーザーがファイルを選択
②ファイル名プロパティを格納
↓ ← ダイアログボックスが閉じる
③ラベルコントロールにファイル名表示

4

ダイアログボックスを表示する前に、［Filter］プロパティでダイアログボックスに表示するファイルの種類を設定しておきます。

```
With CommonDialog1
    .Filter = "すべてのファイル (*.*)|*.*"
```

［Filter］プロパティの値には、実際のフィルタ記号とその説明文を設定できます。上記の例では、「|」の前までが説明文で、後ろが実際に表示させるファイルタイプの記号になります。ダイアログボックスの初期化設定が終了したら、「ShowOpen」メソッドで［ファイルを開く］ダイアログボックスを表示します。

```
.ShowOpen
```

なお、ここでは、このあといくつかのプロパティにアクセスするので、Withステートメントを使ってオブジェクト名の記述を簡素にしています。

5

ダイアログボックスが表示され、ユーザーによってファイルが選択されると、［filename］プロパティにそのファイル名がフルパスで格納されます。そこで、この値を参照してプログラムの処理に利用します。ここでは、［filename］プロパティの値を変数GetNameに格納し、ラベルコントロールの［Caption］プロパティにその変数と文字列を設定してファイル名を表示します。

```
    GetName = .filename
End With
Label1.Caption = "選択したファイル名は" & GetName
```

［ファイルを開く］ダイアログボックス

6

「フォントの設定」ボタンでは、「フォントの指定」ダイアログボックスを使用します。メソッドは「ShowFont」を使用し、ユーザーによって選択されたフォント情報は、以下のプロパティに格納されます。

プロパティ	内容	値
FontName	フォント名	フォント名
FontSize	フォントサイズ	サイズ（ポイント単位）
FontBold	ボールド（太字）	Boolean
FontItalic	イタリック（斜体）	Boolean
FontStrikethru	取り消し線	Boolean
FontUnderline	下線	Boolean

7 ［フォントの指定］ダイアログボックスを使う場合は、ShowFontメソッドを実行する前に、［Flags］プロパティにダイアログボックスに表示するフォントの種類をセットします。このプロパティを設定せずにShowFontメソッドを実行すると、「フォントがインストールされていません」というメッセージがでて、ダイアログボックスが表示されません。プロパティの値には、次のいずれかの定数を指定します。

［フォントの指定］ボタンのプロシージャでは、［Flags］プロパティに「cdlCFBoth」を設定し、ShowFontメソッドを実行します。

定数	説明
cdlCFBoth	プリンタフォントおよびスクリーンフォントの一覧を表示します。
cdlCFPrinterFonts	プリンタがサポートするフォントだけを表示します。
cdlCFScreenFonts	システムでサポートされているスクリーン フォントだけを表示します。

8 そして、ユーザーが選択したフォント名とサイズを、［FontName］と［FontSize］プロパティから取得します。

```
With CommonDialog1
    .Flags = cdlCFBoth
    .ShowFont
    FName = .FontName
    FSize = .FontSize
End With
```

COLUMN ・・・・・

Withステートメント

Withに続いて指定したオブジェクト名を、以降のコードでは省略して式を書くことができます。1つのオブジェクトに対して複数のプロパティやメソッドを連続して実行する場合に使用すると便利です。必ずWith...End Withで記述します。下の左のコードは右のように書いた場合と同様です。

```
With CommonDialog1
    .Flags = cdlCFBoth
    .ShowFont
End With
```

→

```
CommonDialog1.Flags = cdlCFBoth
CommonDialog1.ShowFont
```

9 取得したフォント名（Fname）は、ラベルコントロールの［Caption］プロパティに代入し、フォントサイズ（Fsize）でラベルコントロールの文字サイズの設定に使用しています。

```
With Label1
    .Caption = "このフォントは" & FName
    .FontSize = FSize
End With
```

10 ［色の設定］ボタンでは、ShowColorメソッドを使って［色の設定］ダイアログボックスを表示します。

ユーザーが選択した色は、コモンダイアログボックスコントロールの［Color］プロパティに格納されますので、これをラベルコントロールの［ForeColor］プロパティに代入し、ラベルコントロールの文字色を設定します。

```
Private Sub Command3_Click()
    CommonDialog1.ShowColor
    Label1.ForeColor = CommonDialog1.Color
End Sub
```

まとめ

コモンダイアログボックスは、あらかじめWindowsで用意されているダイアログボックスなので、自分でユーザーインターフェイスを組み立てる必要がなく、使い方も簡単です。目的に合ったメソッドとプロパティを使うのがポイントです。

練習問題

Q1　Lesson22で作成したような、テキストファイルを読み込んで、テキストボックスコントロールに表示するプログラムを、コモンダイアログボックスコントロールを使用して作成してください。

解答は巻末に→

ワンポイントアドバイス

コモンダイアログボックスは、あくまでもWindows共通のユーザーインターフェイスを提供するコントロールで、ユーザーの選択内容をプロパティから受け取った後の処理は、自分で組み立てなければなりません。例えば［フォントの指定］ダイアログボックスでフォントを指定しても、その結果をどのコントロールに反映させるのかを記述しなければ、何も起こらないので注意しましょう。

Lesson 27

第7日 2時限目

いろいろなテクニック

Lesson27.vbp
VB6CD
Lesson27

×Learning Edition

マルチメディアを使う

マルチメディアコントロールのプロパティやメソッドの使い方を覚えます。

今回作成するプログラム

前のトラック（Prev）
次のトラック（Next）
再生（Play）
一時停止（Pause）
Form1
ボタンをクリックすると、サウンドファイルを再生、一時停止、停止できる
停止（Stop）
早送り（Step）
巻き戻し（Back）

POINT

マルチメディアコントロールを使って、サウンドファイルやビデオ、MIDIファイルの再生を行うことができます。

制御できるメディアは以下の通りです。

●オーディオボード
●MIDIシーケンサ
●CD-ROMドライブ
●オーディオCDプレーヤー
●ビデオディスクプレーヤー
●ビデオテープレコーダー
●ビデオテーププレーヤー

ここでは、サウンドファイルを例題にして、マルチメディアコントロールの使い方を紹介します。

コントロールとプロパティの設定

※新しいプロジェクトとフォームを用意します。

1 ツールボックスをマウスの右ボタンでクリックし、メニューから［コンポーネント］を選ぶ

コンポーネント(O)...
タブの追加(A)...
ドッキング可能(K)
非表示(H)

1 ここをクリック

2 ［コントロール］タブで［Microsoft Multimedia Control 6.0］をチェックする

コンポーネント
コントロール｜デザイナ｜挿入可能なオブジェクト
Microsoft Grid Control
Microsoft Hierarchical FlexGrid Control 6.0 (OLEDB)
Microsoft Internet Controls
Microsoft Internet Transfer Control 6.0
Microsoft MAPI Controls 6.0
Microsoft Masked Edit Control 6.0
Microsoft Multimedia Control 6.0
Microsoft MultiMedia DTCs
Microsoft Outline Control
Microsoft PictureClip Control 6.0
Microsoft RemoteData Control 6.0
参照(B)...
選択された項目のみ(S)
Microsoft Multimedia Control 6.0
場所　C:¥WINNT¥System32¥MCI32.OCX
OK　キャンセル　適用(A)

1 ここをチェック

2 ここをクリック

3 ツールボックスに［MMControl］ボタンが追加される

MMControl

追加された［MMControl］ボタン

コンポーネント

コンポーネントとは、プログラムのコードを使って、プログラマーが一貫した方法で利用できるプログラム用「部品」の総称です。

コモンダイアログコントロールやマルチメディアコントロールは「コントロール」という、外部ファイルとして用意されいるコンポーネントの1つの種類です。

ちなみにコモンダイアログコントロールのファイル名は「COMDLG32.OCX」で通常は「Windows¥System」フォルダにインストールされています。

マルチメディアコントロール

マルチメディアコントロールは、Visual Basicの初期設定では、ツールボックスに組み込まれていません。使用する際に、ツールパレットにコントロールを組み込む必要があります。

また、このコントロールはLearning Editionには含まれていません。Professional Edition以上が必要です。

マルチメディアコントロールの配置

配置するとメディアを操作するための操作パネルが表示されます。

これらの操作ボタンは、プロパティを使って1つずつ表示・非表示や有効・無効の設定が行えるので、最初にプロパティウィンドウで不要なボタンを非表示にしておきます。

不要なボタンの削除

このプログラムでは、サウンドファイル（Wavファイル）を再生するので、イジェクトボタンと録音ボタンを無効にします。

コントロールを選択

この操作を行う前に、作成したマルチメディアコントロールが選択されていることを確認してください。

4 ツールボックスで［MMControl］をクリックし、フォーム上にマルチメディアボタンを配置する

Form1

1 ここをクリックして…

2 そのままここまでドラッグ

5 プロパティウィンドウで［EjectVisible］プロパティの値を「False」に設定する

MMControl1 MMControl

全体 | 項目別

DragMode	0 - vbManual
EjectEnabled	False
EjectVisible	False
Enabled	True
FileName	False
Frames	1

1 ここをクリック

2 ここをクリック

3 ここをクリック

6 プロパティウィンドウで［RecordVisible プロパティの値を「False」に設定する

MMControl1 MMControl

全体 | 項目別

PlayEnabled	False
PlayVisible	True
PrevEnabled	False
PrevVisible	True
RecordEnabled	False
RecordMode	1 - mciRecordOverwrite
RecordVisible	False
Shareable	True
	False

RecordVisible

1 ここをクリック

2 ここをクリック

3 ここをクリック

7 操作パネルのボタンが変更される

コードの記述

Form1:Form-Load

Project1 - Form1 (ｺｰﾄﾞ)

Form | Load

```
Private Sub Form_Load()
    MMControl1.DeviceType = "WaveAudio"
    MMControl1.filename = App.Path & "¥music.wav"
    MMControl1.Notify = False
    MMControl1.Wait = True
    MMControl1.Shareable = False

    MMControl1.Command = "Open"
End Sub
```

Form1:Form-Unload

Project1 - Form1 (ｺｰﾄﾞ)

Form | Unload

```
Private Sub Form_Unload(Cancel As Integer)
    MMControl1.Command = "Close"
End Sub
```

解説

1　マルチメディアコントロールは、いろいろなタイプのメディアを操作できるコントロールです。そのために、どのタイプのメディアを使うのか、メディアのファイル名とともにあらかじめ設定する必要があります。

メディアのタイプは、コントロールの［DeviceType］プロパティに、専用の名前を文字列でセットします。

<代表的なメディアと［DeviceType］プロパティ値（一部）>

プロパティの値	メディアタイプ
AVIVideo	ビデオファイル（*.avi）
CDAudio	オーディオCD（*.cda）
Sequencer	MIDIファイル（*.mid）
WaveAudio	サウンドファイル（*.wav）

プロパティの設定は、プログラムデザイン時にプロパティウィンドウで行えますが、コードから設定する場合はフォームのLoadイベントプロシージャで行います。

```
MMControl1.DeviceType = "WaveAudio"
```

2 メディアのタイプを設定したら、使用するファイルを［FileName］プロパティに設定します。ここでは、プロジェクトと同じフォルダ内にある「music.wav」を指定します。再生ファイルは、プロパティウィンドウで指定することもできます。

```
MMControl1.filename = App.Path & "¥music.wav"
```

3 ファイル名を設定したら、ファイルを開く処理を行います。これは、［Command］プロパティに「Open」というコマンド名を設定することで行われます。

このOpenコマンドは、事前にいくつかのプロパティを設定するように指定されています。それらが、［Notify］、［Wait］、［Shareble］プロパティで、値の指定方法も決められています（それぞれの意味については、ここでは特に説明しませんので、Visual Basicのオンラインヘルプを参照してください）。

指定されているプロパティに値をセットしたら、［Command］プロパティに「Open」を設定します。これで、マルチメディアコントロールの操作ボタンで、サウンドファイルを再生できます。

```
MMControl1.Notify = False
MMControl1.Wait = True
MMControl1.Shareable = False

MMControl1.Command = "Open"
```

4 一度ファイルを開いたので、プログラム終了時に必ず使用したデバイスを閉じる処理が必要です。この処理は、プログラム終了時に実行させるので、フォームの「Unload」イベントプロシージャに作成します。

デバイスを閉じるには、マルチメディアコントロールの［Command］プロパティに、「Close」コマンドを設定します。

```
Private Sub Form_Unload(Cancel As Integer)
    MMControl1.Command = "Close"
End Sub
```

5 これで、マルチメディアコントロールの設定が完了です。プログラムを実行させ、［再生］ボタンを押せばWaveファイルが再生されます。なお、続けて再生する場合は、［前のトラック］ボタンを押してから［再生］ボタンを押してください。

<代表的なコマンドと設定プロパティ>

コマンド	内容	プロパティ
Open	デバイスを開きます。	Notify (False) Wait (True) Shareable (False) DeviceType FileName
Close	デバイスを閉じます。	Notify (False) Wait (True)
Play	デバイスを再生します。	Notify (True) Wait (False) From To
Pause	再生または録音や録画を一時停止します。	Notify (False) Wait (True)
Stop	再生または録音や録画を停止します。	Notify (False) Wait (True)
Eject	メディアをイジェクトします。	Notify (False) Wait (True)
Sound	音を鳴らします。	Notify (False) Wait (False) FileName

まとめ

マルチメディアコントロールを使用する場合は、必ず使用するデバイスタイプとファイルの種類を一致させてください。

また、使用するファイル名は、フルパスで記述します。実行するプロジェクトと同じフォルダに入れておく場合は、説明した App オブジェクトの［Path］プロパティなどと組み合わせておくといいでしょう。

練習問題

Q1　「Lesson27.vbp」のプログラムを改修して、MIDI ファイルを再生できるようにして下さい。MIDI ファイルは、「Q&A」フォルダにある「Md001.mid」を使います。

解答は巻末に→

Lesson 28

第７日
２時限目

いろいろなテクニック

Lesson28.vbp
VB6CD
Lesson28

×Learning Edition

コードでサウンドファイルを再生する

コードからサウンドファイルを再生する方法を覚えます。

● 今回作成するプログラム

Form1
3回再生

プロジェクトを実行するとコントロールが消える

Form1
3回再生

ボタンを押すとサウンドファイルが３回再生される

POINT

今度は、サウンドファイルを、マルチメディアコントロールの操作ボタンを使わずに、コードで再生する方法を紹介します。プログラムの中でサウンドを鳴らしたい場合などは、マルチメディアコントロールの操作パネルを非表示にして、機能だけを利用することができます。その場合は、再生操作などをすべてコードで行います。

このプログラムは、コマンドボタンを押すと、サウンドファイルを３回連続で再生します。

コントロールとプロパティの設定

※新しいプロジェクトとフォームを用意します。

1 マルチメディアコントロールをフォームに配置する

2 プロパティウィンドウで［Visible］プロパティの値を「False」に設定する

1 ここをクリック
2 ここをクリック
3 ここをクリック

3 コマンドボタンを配置する

1 ここをクリックして…
2 そのままここまでドラッグ

4 プロパティウィンドウで［Caption］プロパティの値を「3回再生」に変更する

(オブジェクト名)	Command1
Appearance	1 - 3D
BackColor	&H8000000F&
Cancel	False
Caption	3回再生
CausesValidation	True

1 ここをダブルクリック
2 値を入力

マルチメディアコントロール

マルチメディアコントロールは、Visual Basicの初期設定では、ツールボックスに組み込まれていません。使用する際に、ツールパレットにコントロールを組み込む必要があります。

マルチメディアコントロールを配置する方法については、Lesson27を参照してください。

コントロールを選択

この操作を行う前に、作成したマルチメディアコントロールが選択されていることを確認してください。

［Visible］プロパティのセルをダブルクリックして、値を切り替えることもできます。

コマンドボタンを選択

この操作を行う前に、作成したコマンドボタンコントロールが選択されていることを確認してください。

コードの記述

Form1:Form-Load

Project1 - Form1 (コード)
Form / Load

```
Private Sub Form_Load()
    MMControl1.DeviceType = "WaveAudio"
    MMControl1.filename = App.Path & "¥music.wav"
    MMControl1.Notify = False
    MMControl1.Wait = True
    MMControl1.Shareable = False

    MMControl1.Command = "Open"
End Sub
```

Form1:Command1-Click

Project1 - Form1 (コード)
Command1 / Click

```
Private Sub Command1_Click()
    Dim i As Integer

    For i = 1 To 3
        MMControl1.Command = "Sound"
    Next
```

Form1:Form-Unload

Project1 - Form1 (コード)
Form / Unload

```
Private Sub Form_Unload(Cancel As Integer)
    MMControl1.Command = "Close"
End Sub
```

ワンポイントアドバイス

実際にプログラムを作成する場合は、プログラムの終了処理を記述するプロシージャの中で、Endステートメントの前に必ずフォームに対しUnloadステートメントを実行してください。そうしないと、フォームのUnloadイベントプロシージャが実行されません。「フォームの有効期間」についてはP.172のコラムを参照してください。

解説

1 マルチメディアコントロールは、［Visible］プロパティをFalseに設定したので、プログラム実行時はフォームに表示されません。したがって操作ボタンが使えないので、ユーザーは一切操作できなくなります。メディアの操作をコードで行う場合は、このようにコントロールを非表示にしておくといいでしょう。

2 フォームのLoadイベントプロシージャで、マルチメディアコントロールの初期化をしておきます。これは、Lesson27のプログラムといっしょです。

3 次に、コマンドボタンのClickイベントプロシージャに、サウンド再生の処理を作成します。

サウンドファイルの再生は、［Command］プロパティに「Sound」コマンドを設定します（Lesson27参照）。これを、For...Nextループで3回実行するようにします。

```
Dim i As Integer

For i = 1 To 3
    MMControl1.Command = "Sound"
Next
```

4 最後に、フォームのUnloadイベントプロシージャに、デバイスを閉じる処理を記述して完成です。プログラムを実行してコマンドボタンを押すと、サウンドファイルが3回再生されます。

まとめ

コードでメディアを操作する場合は、［Command］プロパティにコマンドをセットして実行します。

なお、「Sound」コマンドは、WAV形式のサウンドファイルの再生に使うコマンドで、MIDIファイルなどでは「Play」コマンドを使用します。

練習問題

Q1 このプログラムを改修して、コマンドボタンが押されるとコモンダイアログボックスの［ファイルを開く］ダイアログボックスが開き、選択したファイルを３回再生できるようにしてください。

解答は巻末に→

Lesson 29

第７日
３時限目

いろいろな
テクニック

Lesson29.vbp
VB6CD
Lesson29

◎Learning Edition

データコントロール

データコントロールの初歩的な使い方を覚えます。また、連結コントロールを操作してデータを表示する方法を知ります。

今回作成するプログラム

Form1

Data1

販売実績

販売月数 5

香水 138

口紅 267

ファンデーション 642

前後のボタンをクリックすると…

月ごとのデータを合計した数値が表示される

本社集計 : テーブル

月	香水	口紅	ファンデーション	マニキュア
1	165	363	636	522
2	63	396	567	504
3	165	603	591	516
4	114	534	693	579
5	138	267	642	468
6	159	264	675	486
7	183	276	654	510
8	147	312	621	507
9	138	456	603	435
10	144	339	597	444
11	156	441	528	417
12	123	462	567	456
(オートナンバー)				

レコード: 1 / 12

使用するデータベースファイル「本社合計 1.mdb」

!!! POINT

データコントロールを使用すると、Access などのデータベースファイルのデータを、プログラムで使用できるようになります。

また、ラベルコントロールなどは、データコントロールと連動できる「連結」コントロールになっているため、プロパティを設定するだけで簡単にレコードへのアクセスができます。

ここでは、Access で作成したデータベースファイル「本社合計 1.mdb」を使って、レコードデータを読み出すプログラムを作成します。「本社合計 1.mdb」は、「Lesson29」フォルダに入っています。

コントロールとプロパティの設定

※新しいプロジェクトとフォームを用意します。

1 ツールボックスの［Data］ボタン をクリックし、フォーム上にデータコントロールを配置する

Form1

Data1

1 ここをクリックして…

2 そのままここまでドラッグ

2 プロパティウィンドウで［DatabaseName］プロパティにデータベースファイルを設定する

ﾌﾟﾛﾊﾟﾃｨｰ Data1

Data1 Data

全体　項目別

BOFAction	0 - 先頭ﾚｺｰﾄﾞ
Caption	Data1
Connect	Access
DatabaseName	...
DefaultCursorType	0 - 既定値のｶｰｿﾙ
DefaultType	2 - Jet

1 ここをクリック

2 ここをクリック

ﾃﾞｰﾀﾍﾞｰｽ名

ﾌｧｲﾙの場所(I): Lesson29

本社合計1.MDB

ﾌｧｲﾙ名(N): 本社合計1.MDB

ﾌｧｲﾙの種類(T): ﾃﾞｰﾀﾍﾞｰｽ (*.mdb)

開く(O)　ｷｬﾝｾﾙ　ﾍﾙﾌﾟ(H)

3 ここをクリック

4 ここをクリック

連結コントロール

データコントロールでデータベースファイルを指定すると、ラベルコントロールなどの他のコントロールからもそのデータベースのテーブルやフィールドに自動的にアクセス可能になります。こういった連動して動作するコントロールを「連結コントロール」と呼んでいます。

ラベル以外にも、チェックボックス、イメージ、テキストボックスなどが連結コントロールになります。詳しくはオンラインヘルプを参照してください。

データ（Data）コントロール

Accessなどのデータベースファイルにアクセスするためのコントロールです。

連結コントロールを使って、レコード間を移動したり、レコードのデータを表示したり操作できます。

DatabaseName

データコントロールのデータソースの名前と場所を設定するプロパティです。

RecordsetType

データコントロールが作成するRecordsetオブジェクトの種類です。数値で設定します。

RecordSource

データコントロールの基になるテーブル、SQLステートメントなどを指定します。ここでは「本社合計1.mdb」内のテーブル「本社集計」を設定しました。

3 [RecordsetType] プロパティの値を「0 - テーブル」に設定する

4 [RecordSource] プロパティでテーブル「本社集計」を選ぶ

5 ツールボックスで [Frame] をクリックし、フォーム上にフレームコントロールを配置する

6 プロパティウィンドウで［Caption］プロパティの値を「販売実績」に変更する

Frame1 Frame	
(オブジェクト名)	Frame1
Appearance	1 - 3D
BackColor	&H8000000F&
BorderStyle	1 - 実線
Caption	販売実績
ClipControls	True

1 ここをダブルクリック

2 値を入力

7 ツールボックスの［Label］ A をクリックし、フォーム上にラベルコントロールを４つ配置する

販売実績
Label1
Label2
Label3
Label4

これぐらいのサイズのラベルを４つ作る

8 各ラベルコントロールの［Caption］と［Alignment］プロパティの値を次のように変更する

コントロール名	[Caption]	[Alignment]
Label1	月数	[1 - 右揃え]
Label2	香水	[1 - 右揃え]
Label3	口紅	[1 - 右揃え]
Label4	ファンデーション	[1 - 右揃え]

販売実績
販売月数
香水
口紅
ファンデーション

キャプションが変更されたラベル

ラベルを選択

この操作を行う前に、作成したラベルコントロールが選択されていることを確認してください。

9 ツールボックスの［Label］ をクリックし、ラベルの隣にラベルコントロールを４つ配置する

10 各ラベルの［BorderStyle］プロパティの値を「1 - 実線」に変更する

11 各ラベルの［DataSource］と［DataField］プロパティの値を次ように変更する

コントロール名	[DataSource]	[DataField]
Label5	Data1	月
Label6	Data1	香水
Label7	Data1	口紅
Label8	Data1	ファンデーション

DataSource

連結するデータコントロールを指定するプロパティです。

DataField

データコントロールが参照しているデータベースのテーブルから、カレントレコードのどのフィールドを参照するのかを指定します。

データコントロールがデータベースファイルを指定していると、そのファイルが使用中の各フィールドが自動的にポップアップメニューに表示されます。

12 完成したレイアウト

Form1

Data1

販売実績

販売月数 Label5

香水 Label6

口紅 Label7

ファンデーション Label8

解説

1 　データコントロールは、［DatabaseName］プロパティで、使用するデータベースファイルを指定します。そして、［RecordSource］プロパティで、使用するテーブルを指定します。

　ラベルコントロールは、［DataSource］［DataField］プロパティでデータコントロールを指定した時点で、そのコントロールと連動して動作する連結コントロールになります。

まとめ

　このプログラムでは、特にコードは作成しません。データコントロールは、フォームに配置して連結コントロールと連動させるだけで、自動的にテーブルとフィールドにアクセスできます。１つのデータコントロールでアクセスできるテーブルは１つだけです。複数のテーブルに同時にアクセスするには、複数のデータベースコントロールを配置して使用します。

練習問題

Q1 　「Lesson29.vbp」では、あらかじめプロパティウィンドウでデータベースファイルを指定しました。しかし、プログラムを他のコンピュータに配布する場合は、ディレクトリ名が変わってしまいます。それを回避するために、プログラム起動時にプログラムと同じフォルダにあるデータベースファイルを使用できるように変更してください。

解答は巻末に→

Lesson 30

第７日
４時限目

いろいろなテクニック

Lesson30.vbp
VB6CD
Lesson30

◎Learning Edition

タブ付きダイアログボックス

タブ付きダイアログボックスの使い方を覚えます。また、各ページに配置したコントロールの操作方法と、フォームから呼び出して使う方法も紹介します。

● 今回作成するプログラム

Form1

タブの表示

1 このボタンをクリックすると…

Form2

ページ1　ページ2　ページ3

閉じる

2 タブ付きダイアログを含むフォームが表示される

POINT

タブ付きダイアログボックスは、Windows 95から採用された新しい組み込みダイアログボックスです。タブをクリックすることで、表示するページを切り替えて使用できます。各ページにはコントロールをいろいろなレイアウトで配置できます。

ここでは、このタブ付きダイアログボックスコントロールを使用する方法を紹介します。

コントロールとプロパティの設定

※新しいプロジェクトとフォームを用意します。

1 ツールボックスの［CommandButton］をクリックし、Form1にコマンドボタンを配置する

Form1

タブの表示

1 ［Caption］プロパティを「タブの表示」に設定

2 ［標準］ツールバーの［フォームモジュールの追加］をクリックし、［フォームモジュール］を選ぶ

Project1 - Microsoft Visual Basic [デザイン]

ファイル(F)　編集(E)　表示(V)　プロジェクト(P)　書式(O)　デバッグ(D)　実行(R)

フォーム モジュール(F)
MDI フォーム モジュール(I)
標準モジュール(M)
クラス モジュール(C)
ユーザー コントロール(U)
プロパティ ページ(R)
ユーザー ドキュメント(D)
WebClass
Data Report
DHTML Page
Data Environment
その他の ActiveX デザイナ(G)...
ファイルの追加(A)...　Ctrl+D

タブの表示

1 ここをクリック

2 ここをクリック

3 ［フォームモジュール］を選択し、［開く］をクリックして、新しいフォーム（Form2）を追加する

フォーム モジュールの追加

新規作成　既存のファイル

フォーム モジュール　データ フォーム ウィザード　バージョン情報　Web ブラウザ　ダイアログ　ログオン ダイアログ　見出し画面　ワンポイント　ODBC ログイン　オプション ダイアログ

開く(O)　キャンセル　ヘルプ(H)

次回からこのダイアログ ボックスを表示しない(U)

1 ここをクリック

2 ここをクリック

次回からこのダイアログボックスを表示しない

［次回からこのダイアログボックスを表示しない］チェックボックスをクリックしてチェックしておくと、次回からこのダイアログボックスが表示されなくなります。

4 ツールボックスをマウスの右ボタンでクリックし、メニューから［コンポーネント］を選ぶ

コンポーネント(O)...
タブの追加(A)...
ドッキング可能(K)
非表示(H)

1 ここをクリック

5 ［コントロール］タブで［Microsoft Tabbed Dialog Control 6.0］をチェックする

コンポーネント
コントロール | デザイナ | 挿入可能なオブジェクト
Microsoft RemoteData Control 6.0
Microsoft Rich Textbox Control 6.0
Microsoft Script Control 1.0
Microsoft Scriptlet Component
Microsoft SysInfo Control 6.0
Microsoft Tabbed Dialog Control 6.0
Microsoft Wallet
Microsoft Windows Common Controls 5.0 (SP2)
Microsoft Windows Common Controls 6.0
Microsoft Windows Common Controls-2 5.0 (SP2)
Microsoft Windows Common Controls-2 6.0
参照(B)...
選択された項目のみ(S)
Microsoft Tabbed Dialog Control 6.0
場所　C:¥WINNT¥System32¥TABCTL32.OCX
OK　キャンセル　適用(A)

1 ここをチェック

2 ここをクリック

6 ツールボックスに［SSTab］ボタンが追加される

SSTab

追加された［SSTab］ボタン

タブ付きダイアログボックスコントロール

タブ付きダイアログボックスコントロールコントロールは、Visual Basicの初期設定では、ツールボックスに組み込まれていません。使用する際に、ツールパレットにコントロールを組み込む必要があります。

SSTab

タブダイアログコントロールのことです。

7 ツールボックスで［SSTab］ボタンをクリックし、Form2上にタブ付きダイアログを配置する

1 ここをクリックして…

2 そのままここまでドラッグ

8 各タブの［Caption］プロパティを次のように変更する

コントロール名	[Caption]
Tab0	ページ1
Tab1	ページ2
Tab2	ページ3

9 ［CommandButton］ をクリックし、［ページ1］タブにコマンドボタンを配置する

1 ［Caption］プロパティを「メッセージの表示」に設定

［ページ2］タブを表示

この操作を行う前に、フォーム上で［ページ2］タブをクリックして、［ページ2］タブを表示してください。

10 ［CommandButton］ をクリックし、［ページ2］タブにコマンドボタンを配置する

Form2

ページ1　ページ2　ページ3

タブを左にする

1 ［Caption］プロパティを「タブを左にする」に設定

［ページ3］タブを表示

この操作を行う前に、フォーム上で［ページ3］タブをクリックして、［ページ3］タブを表示してください。

11 ツールボックスの［Image］をクリックし、［ページ3］タブにイメージコントロールを配置する

Form2

ページ1　ページ2　ページ3

1 ここをクリックして…

2 そのままここまでドラッグ

COLUMN

ActiveX コントロールの配布

「タブ付きダイアログボックスコントロール」など、標準コントロールでないコントロールを組み込んだプログラムを作成して、他のコンピュータに配布する場合は、これらのコントロールもいっしょに配布する必要があります。

どのコントロールがどういうファイル名になっているのかは、オンラインヘルプを見ればわかりますが、Lesson36で解説する「ディストリビューションウィザード」を使って配布キットを作成すれば、間違いなくインストールできます。

12 プロパティウィンドウで［Picture］プロパティにイメージファイルを設定する

プロパティ - Image1

Image1 Image

全体 | 項目別

MousePointer	0 - 既定値
OLEDragMode	0 - 手動
OLEDropMode	0 - なし
Picture	(なし)
Stretch	True
Tag	

1 ここをクリック

2 ここをクリック

ピクチャの読み込み

ファイルの場所(I): Windows

Hlpstep3.gif　Winlogo.gif　Winupd.ico　エジプト.bmp　しゃぼん玉.bmp　セットアップ.bmp　トライアングル.bmp　バンブー.bmp　ピンストライプ.bmp　ブラック フロア.bmp　ブルー ウェーブ.bmp　ブルー リベット.bmp　ブルー リング.bmp　レッド タイル.bmp　花見.bmp　千鳥格子.bmp

ファイル名(N): 花見.bmp

ファイルの種類(T): すべてのピクチャ ファイル

開く(O)　キャンセル　ヘルプ(H)

3 ここをクリック

4 ここをクリック

使用するイメージファイル

「C:¥Windows」のフォルダにある「花見.bmp」を使います。

13 挿入したイメージのサイズを変更する

Form2

ページ1　ページ2　ページ3

1 ここをドラッグ

イメージのサイズ変更

イメージの周囲に表示されているサイズ変更ハンドル（黒四角）をドラッグします。

値の設定

[Stretch] プロパティのセルをダブルクリックして、値を切り替えることもできます。

14 プロパティウィンドウで [Stretch] プロパティの値を「True」に設定する

15 [CommandButton] をクリックし、Form2 にコマンドボタンを配置する

16 プロパティウィンドウで [Caption] プロパティの値を「閉じる」に設定する

コードの記述

Form1:Command1-Click

Project1 - Form1 (ｺｰﾄﾞ)

Command1 | Click

```
Private Sub Command1_Click()
    Form2.Show
End Sub
```

Form2:Command1-Click

Project1 - Form2 (ｺｰﾄﾞ)

Command1 | Click

```
Private Sub Command1_Click()
    MsgBox "ページ１のボタンです"
End Sub
```

Form2:Command2-Click

Project1 - Form2 (ｺｰﾄﾞ)

Command2 | Click

```
Private Sub Command2_Click()
    SSTab1.TabOrientation = ssTabOrientationLeft
End Sub
```

Form2:Command3-Click

Project1 - Form2 (ｺｰﾄﾞ)

Command3 | Click

```
Private Sub Command3_Click()
    Form2.Hide
    SSTab1.TabOrientation = ssTabOrientationTop
End Sub
```

解説

1 タブ付きダイアログボックスに配置したコントロールは、フォームに配置したときと同じように扱います。つまり、どのページであっても、コントロールのオブジェクト名でプロパティやメソッドが使えます。

まず、Form1のコマンドボタンのClickイベントプロシージャで、Form2を表示します。

```
Form2.Show
```

2 Form2に配置したタブ付きダイアログボックスの、ページ1にあるコマンドボタンでは、メッセージボックスを表示します。

```
MsgBox "ページ1のボタンです"
```

3 ページ2のコマンドボタンでは、タブ付きダイアログボックスの形式を変更します。［TabOrientation］プロパティは、タブの付く方向を設定するプロパティです。ここでは、左側にタブが付くタイプに変更しています。同様に、「ssTabOrientationTop」、「ssTabOrientationButtom」、「ssTabOrientationRight」で、タブの方向を上下左右に設定できます。

```
SSTab1.TabOrientation = ssTabOrientationLeft
```

4 Form2に配置したコマンドボタンは、Form2を閉じるために使っています。

```
Form2.Hide
SSTab1.TabOrientation = ssTabOrientationTop
```

まとめ

タブ付きダイアログボックスは、1つのコントロールに何ページも設定できるので、フォームをいくつも配置するよりも、プロジェクトの管理がしやすくなります。

また、プログラムを使うユーザーにとっても、操作がわかりやすくなるメリットがあります。

Q 練習問題

Q1

タブを６つ持ったダイアログボックスを作成してください。
タブは上に取り付くタイプで、３つで一列になるようにします。

解答は巻末に→

! ワンポイントアドバイス

タブ付きダイアログボックスで使えるプロパティを紹介しておきましょう。

プロパティ	目的
Font	タブの文字属性を設定します。
ForeColor	タブの文字属性を設定します。
Picture	タブにビットマップを表示します。
Style	ダイアログボックスのスタイルを設定します。
TabHeight	タブの高さを設定します。
Tabs	タブの数を設定します。
TabsPerRow	一列のタブ数を設定します。

COLUMN ・・・・・

タブ付きダイアログボックスのタイプ

タブ付きダイアログボックスは、［TabOrientation］プロパティで、タブの位置をダイアログボックスの上下左右のどこに付けるかを設定できます。また、［Style］プロパティでは、タブの形状を変更できます。［Picture］プロパティは、タブにイメージファイルを設定します。

プロパティウィンドウにある「プロパティページ」を使うと、タブ付きダイアログボックスの形状を一括して設定できます。

Lesson 31

第７日
５時限目

いろいろなテクニック

Lesson31.vbp
VB6CD
Lesson31

◎Learning Edition

フォームテンプレートの活用

フォームテンプレート「見出し画面」の利用方法と表示方法を覚えます。

● 今回作成するプログラム

Form1

バージョン情報

1 このボタンをクリックすると…

ライセンス先: Haruka Seto. 1998.

VB Lab TEchnology

Lesson31

Windows98対応

バージョン 1.0.0

著作権: 瀬戸 遥

警告: このプログラムを無断で販売・複製・再配布することを固く禁じます。

2 バージョン情報が表示される

POINT

Visual Basic 6.0では、プログラム起動時のロゴやアバウトダイアログボックスなど、いくつかのフォームをテンプレートで用意しています。このテンプレートを使うと、最初からフォームを組み立てるよりもずっと楽に作成できます。

ここでは、フォームのテンプレートの１つ「見出し画面」を使って、フォームを作成する方法を紹介します。

コントロールとプロパティの設定

※新しいプロジェクトとフォームを用意します。

1 ツールボックスの［CommandButton］をクリックし、フォーム上にコマンドボタンを配置する

1 ［Caption］プロパティを「バージョン情報」に設定

2 ［標準］ツールバーから［フォームモジュール］を選ぶ

1 ここをクリック

2 ここをクリック

3 ［新規作成］タブで［見出し画面］を選ぶ

1 ここをクリック

2 ここをクリック

次回からこのダイアログボックスを表示しない

［次回からこのダイアログボックスを表示しない］チェックボックスをクリックしてチェックしておくと、次回からこのダイアログボックスが表示されなくなります。

4 あらかじめ製品のバージョン情報のサンプルが設定されているフォームが挿入される

オブジェクト名は「frmSplash」になっている

テンプレート情報の変更

フォーム上の情報はラベルコントロールで作成されています。情報を変更するには、変更するラベルを選択し、プロパティウィンドウで［Caption］プロパティの値を変更します。

5 各ラベルの［Caption］プロパティの値を次のように変更する

元のラベル	変更後のラベル
制作会社名	VB Lab TEchnology
プラットフォーム	Windows98 対応
著作権	著作権：瀬戸 遥
会社名	なし
警告	警告：このプログラムを無断で販売・複製・再配布することを固く禁じます。
ライセンス先	ライセンス先：Haruka Seto.　1998

6 完成した「見出し画面」フォームのレイアウト

7 ［プロジェクト］メニューから［プロジェクト１のプロパティ］を選ぶ

1 ここをクリック

2 ここをクリック

8 ［プロジェクトのプロパティ］ウィンドウの［スタートアップの設定］から［frmSplash］を選ぶ

1 ［frmSplash］を選択

2 ［OK］をクリック

コードの記述

frmSplash:From-Unload

Project1 - frmSplash (ｺｰﾄﾞ)

Form | Unload

```
Private Sub Form_Unload(Cancel As Integer)
    Form1.Show
End Sub
```

Form1:Command1-Click

Project1 - Form1 (ｺｰﾄﾞ)

Command1 | Click

```
Private Sub Command1_Click()
    frmSplash.Show
End Sub
```

解説

1 「見出し画面」は、プログラム起動時に最初に表示されるフォームです。しかし、Form1以外にフォームを追加していますので、このままでは「見出し画面」は最初に表示されません。そこで、「見出し画面」が表示された後に、フォームが表示されるようにします。

2 Visual Basicの［プロジェクト］メニューから［プロジェクト1のプロパティ］を選びます。［プロジェクトプロパティ］というウィンドウが表示されるので、［スタートアップの設定］で［frmSplash］を選びます。

これで、「見出し画面」フォームが最初に表示されるようになります。

Project1 - ﾌﾟﾛｼﾞｪｸﾄ ﾌﾟﾛﾊﾟﾃｨ

全般 | 実行可能ファイルの作成 | ｺﾝﾊﾟｲﾙ | ｺﾝﾎﾟｰﾈﾝﾄ | ﾃﾞﾊﾞｯｸﾞ

ﾌﾟﾛｼﾞｪｸﾄの種類(T): 標準 EXE
ｽﾀｰﾄｱｯﾌﾟの設定(S): frmSplash
ﾌﾟﾛｼﾞｪｸﾄ名(N): Project1
ﾍﾙﾌﾟ ﾌｧｲﾙ名(H):
ﾌﾟﾛｼﾞｪｸﾄ ﾍﾙﾌﾟ ｺﾝﾃｷｽﾄ番号(I): 0
ﾌﾟﾛｼﾞｪｸﾄの説明(P):
対話型ｲﾝﾀｰﾌｪｲｽの抑制(E)
ActiveX ｺﾝﾄﾛｰﾙのｱｯﾌﾟｸﾞﾚｰﾄﾞ(U)
常時に保持(Y)
ﾗｲｾﾝｽ ｷｰを要求する(L)
ｽﾚｯﾄﾞ ﾓﾃﾞﾙ(M)
ｵﾌﾞｼﾞｪｸﾄごとのｽﾚｯﾄﾞ(P)
ｽﾚｯﾄﾞ ﾌﾟｰﾙ(O) 1 ｽﾚｯﾄﾞ
OK | キャンセル | ヘルプ

［プロジェクトプロパティ］ウィンドウ

3　しかし、このままだと「見出し画面」フォームだけしか表示されません。そこで、「見出し画面」フォームが消えると、メインのフォームが表示されるように、frmSplashのUnloadイベントプロシージャで、Form1を表示する処理を記述します。

なお、「見出し画面」フォームは、キーボードが押されるか、クリックされると自分を閉じる処理があらかじめ組み込まれています。

```
Private Sub Form_Unload(Cancel As Integer)
    Form1.Show
End Sub
```

4　プログラムを実行させてください。最初に、見出し画面が表示され、クリックするとこれが閉じて、メインのフォームが表示されます。

5　今度は、フォームと同時に見出し画面が表示されるようにしてみましょう。
まず、今作成した、frmSplashのUnloadイベントプロシージャを削除します。

6　次に、［プロジェクト］メニューの［標準モジュールの追加］で標準モジュールを挿入します。

［標準モジュールの追加］を選択

［標準モジュール］を選択

7 ［Module1］のコードに、「Sub Main」というプロシージャを作成し、2つのフォームを表示するコードを記述します。
このとき、一番上に表示されるフォームを最後に記述します。

Module1:(General)-Main

```
Sub Main()
    Form1.Show
    frmSplash.Show
End SUb
```

8 Visual Basicの［プロジェクト］メニューから［プロジェクト1のプロパティ］を選び、［プロジェクトプロパティ］ダイアログボックスで［スタートアップの設定］ボックスの設定を［Sub Main］にします。

［Sub Main］に設定

9 プログラムを実行します。Form1と見出し画面の両方が表示され、見出し画面が最上面にきます。
この状態では、両方のフォームが表示され、メインのフォームの操作もできます。

完成版は「Lesson312.vbp」としてCD-ROMに収録

まとめ

「見出し画面」は、プログラムの製品名やバージョン情報を表示するのに使います。プログラムのライセンス表示の意味もありますので、有効に使ってください。

この見出し画面は、自分でも作成できますが、最初からレイアウトされているテンプレートを使うと、作業の手間が省けます。

テンプレートには、他にもいろいろと用意されていますから、目的に合ったテンプレートを選んで、活用するといいでしょう。

練習問題

Q1　「Lesson31.vbp」に、テンプレートにある「バージョン情報」を追加し、Form1に配置したコマンドボタンを押すと表示されるようにしてください。

解答は巻末に→

ワンポイントアドバイス

この「見出し画面」フォームにある「製品名」と「バージョン」は、frmSplash内のコードによってプログラム動作時に、自動的に表示されるようになっていますので、書き換えても実行時は反映されません。「製品名」は、プロジェクトファイルの名前が、バージョンはプロジェクトのバージョン情報（プロジェクトプロパティウィンドウで設定）が使われます。それ以外の文字は、ラベルコントロールのCaptionプロパティを変更して、好きな文字に変えて使用します。

また、イメージも、初期設定ではVisual Basicのアイコンが割り当てられていますから、自分の好きなイメージに入れ替えてください。

ワンポイントアドバイス

「フォームレイアウト」ウィンドウを使用すると、デスクトップに表示されるフォームの位置を、自由に設定できます。ただし、使用されるWindowsの画面解像度によって、表示位置が多少ずれる場合がありますので、注意してください。

Let's VBSchool!!

第8日 (Lesson32～Lesson34)

デバッグとエラー処理

プログラム作成で欠かせないのが、デバッグ作業です。Visual Basicでは、変数の内容を見る「ウォッチ」や、プログラムを1行ずつ実行できる「ステップ実行」、コードの途中に設定して一気にそこまで実行させる「ブレイクポイント」など、プログラムの動作をチェックする機能を装備しています。

今日のレッスンでは、これらのデバッグ機能の使い方を説明します。また、プログラムの途中でエラーになった場合に、プログラムを停止させずに処理を続行させる「エラー処理」の使い方を紹介します。

今日のレッスンで解説する項目

- ❏ プログラムを1行ずつ実行させる「ステップ実行」の操作
- ❏ 特定の変数の値の変化を見る「ウォッチ」操作
- ❏ ブレークポイントの設定と操作方法
- ❏「OnError」ステートメントを使ったエラー処理の組み立て方

Lesson 32

第８日
１時限目

デバッグと
エラー処理

Lesson32.vbp
VB6CD
Lesson32

◎Learning Edition

変数のウォッチを使う

プログラムを１行ずつ実行させる、「ステップ実行」の操作を覚えます。また、ステップ実行をしながら、特定の変数の値の変化を見る「ウォッチ」操作を身に付けます。

今回作成するプログラム

Form1
円の描画

1 このボタンをクリックすると…

Form1
円の描画

2 ５つの円が描画される

POINT

For...Next ループを使って、円を５つフォームに描画する簡単なプログラムを作り、ステップ実行と変数のウォッチを使ってみます。

コントロールとプロパティの設定

※新しいプロジェクトとフォームを用意します。

1 ツールボックスで［CommandButton］をクリックし、フォーム上にコマンドボタンを配置する

Form1

Command1

1 ここをクリックして…

2 そのままここまでドラッグ

2 プロパティウィンドウで［Caption］プロパティの値を「円の描画」に変更する

Command1 CommandButton

全体 | 項目別

(ｵﾌﾞｼﾞｪｸﾄ名)	Command1
Appearance	1 - 3D
BackColor	&H8000000F&
Cancel	False
Caption	円の描画
CausesValidation	True

1 ここをダブルクリック

2 「円の描画」と入力

エラー

プログラムの誤りのこと。不正な処理が起きたときに発生します。

バグ

プログラムの不具合のこと。バグがあるとプログラムが正常に動作しません。

デバッグ

「バグ」を取り除く作業をデバッグといいます。

ウォッチ

プログラムの実行中に値の変化を監視する機能。デバッグ作業に有効です。

ステップ実行

プログラムを1行ずつ実行すること。Visual Basicには「ステップイン」、「ステップオーバー」、「ステップアウト」の3種類が用意されています。

コードの記述

Form1:Command1-Click

Project1 - Form1 (ｺｰﾄﾞ)

Command1　Click

```
Private Sub Command1_Click()
    Dim x As Integer, y As Integer
    Dim i As Integer

    x = 500
    y = 1000
    For i = 0 To 4
        Circle (x, y), 400
        x = x + 800
    Next
End Sub
```

解説

1 プログラムのコードそのものは、簡単なコードです。「Circle」メソッドを使って、フォームに円を描画します。「(x、y)」は円の中心位置座標、「400」は円の半径を設定しています。このとき、For...Nextループで、5つの円を連続で描画します。円が重ならないように、描くたびに右にずらしていきます（x+800）。

2 まず、円の中心の座標「x」の値がどう変化するのか、変数の値を見られるようにします。

Visual Basicの［デバッグ］メニューから［ウォッチ式の追加］を選びます。

1 ここを選択

3 ［ウォッチ式の追加］ダイアログボックスが表示されますので、［式］ボックスに変数名「x」と入力します。そして、［OK］ボタンをクリックします。

1 「x」を入力

4 Visual Basicの画面下部に、［ウォッチ］というウィンドウが表示されます。ここに、変数の値が表示され、プログラムを実行しながら変数の中身を知ることができます。

式	値	型	対象
x	<対象範囲外>	Empty	Form1.Command1_Click

5　今度は、デバッグ機能の「ステップイン」を使って、コードを1行ずつ実行していきます。

まず、Visual Basicの［デバッグ］メニューから［ステップイン］（ショートカットキーはF8）を選びます。

1 ここを選択

6　プログラムの実行が開始され、フォームが表示されます（実行中のプログラムとVisual Basicのウィンドウの両方が見えるようにに画面をレイアウトしてください）。

フォームの［円の描画］ボタンをクリックします。コードウィンドウのプロシージャ名が黄色で範囲選択されます。

```
Private Sub Command1_Click()
    Dim x As Integer, y As Integer
    Dim i As Integer

    x = 500
    y = 1000
    For i = 0 To 4
        Circle (x, y),
        x = x + 800
    Next
End Sub
```

1 ここをクリックすると…

ここが黄色に反転する

7　F8キーを押すと、黄色の反転表示が次の行に移動します。そのまま、Forループが1回行われる（「Next」行が黄色に反転する）まで、F8キーを押し続けます。

現在実行中の行が黄色の反転表示になる

8 一方、ウォッチウィンドウにある変数「x」の値は、「x=500」にきた時点では「500」と表示されています。

これが、ループが実行され、「x = x + 800」の行を過ぎると「1300」に変化します。すなわち、「x = 500 + 800」という式が実行されて、変数の値が変化したのです。

プログラムの実行画面では、フォームに円が描画されています。

ウォッチ

式	値	型	対象
x	1300	Integer	Form1.Command1_Click

変数の値が表示されている

このウィンドウが表示されないときは［表示］メニューから［ウォッチウィンドウ］を選択

Form1

円の描画

フォームに円が描画された

9 F8キーを押し続け、もう一度ループを実行します。変数「x」の値がまた変化し、フォームにもう1つ円が描画されます。さらに、F8キーを押し続け、「End Sub」の所に矢印がきたら終了です。また、［デバッグ］メニューから［ステップアウト］を選択するとステップイン機能が終了し、通常の実行モードになります。

Project1 - Form1 (コード)

Command1　Click

```
Private Sub Command1_Click()
    Dim x As Integer, y As Integer
    Dim i As Integer

    x = 500
    y = 1000
    For i = 0 To 4
        Circle (x, y), 400
        x = x + 800
    Next
End Sub
```

ウォッチ

式	値	型	対象
x	3700	Integer	Form1.Command1_Click

変数の値が「3700」に変化する

```
Project1 - Form1 (コード)
Command1    Click
Private Sub Command1_Click()
    Dim x As Integer, y As Integer
    Dim i As Integer

    x = 500
    y = 1000
    For i = 0 To 4
        Circle (x, y), 400
        x = x + 800
    Next
End Sub
```

ここにきたら終了

10　このようにして、ステップイン機能を使えばプログラムを1行ずつ実行し、動作を確認することができます。

また、ウォッチ機能を併用することで、プログラムを実行しながら変数の値の変化を見ることができるので、プログラムが設計どおり動作しているのかどうかを確認できます。

まとめ

大規模なプロジェクトになればなるほど、プログラムの動作確認は大変な作業になってきます。

ステップ実行は、プログラムの動作確認に使うと、たいへん便利な機能です。変数の値のチェックも重要です。設計どおりの値になっているのかどうかは、このウォッチ機能を使えば簡単に確認できます。

練習問題

Q1　このプログラムに、ループのカウンタ変数「i」の値が見えるように、ウォッチウィンドウに追加してください。

解答は巻末に→

ワンポイントアドバイス

ウィンドウが重なったときに、フォームの円がちゃんと描画されない場合は、Form1の［AutoRedraw］プロパティを「True」に設定してください。こうすると、フォームが他のウィンドウに隠れても、ちゃんと元どおりに描画し直されます。

Lesson 33

第８日 ２時限目

デバッグとエラー処理

Lesson33.vbp
VB6CD
Lesson33

◎Learning Edition

ブレークポイントを使う

ブレークポイントの設定と操作方法を覚えます。
今回は、特にプログラムは作成しません。すでに作成されている「Q&A-25.vbp」を使用して、ブレークポイントの方法を紹介します。

今回作成するプログラム

1 プログラムを登録しようとすると…

2 ここでブレークする

POINT

プログラムのある行にポイントを設け、そこまでは通常どおりに実行させ、そこからステップ実行させることができます。このポイントを、「ブレークポイント」といいます。比較的長いコードで、場所を特定してステップ実行させたい場合に使うと便利です。

解 説

1 まず、付録ディスクの「Q&A」フォルダに収録されている「Q&A-25.vbp」を読み込みます。そして、「Setup」フォームのコードウィンドウを表示します。

```
Project1 - Setup (コード)
(General)                SetupApp
Dim i As Integer

Private Sub AddProgram_Click()
    Call SetupApp
End Sub

Private Sub cxsetup_Click()
    Setup.Hide
End Sub

Private Sub Drive1_Change()
    On Error GoTo FAIL
    Dir1.Path = Drive1.Drive
    Exit Sub

FAIL:
    MsgBox Error(Err.Number)
End Sub

Private Sub Dir1_Change()
    File1.Path = Dir1.Path
End Sub

Private Sub File1_DblClick()
```

コードウィンドウを表示する

2 次に、「Sub SetupApp()」プロシージャに移動します。
ここにある、「SaveSetting」の記述の、左端をマウスでクリックすると、●印がウィンドウに付きます。
これで、この行にブレークポイントが設定されました。

```
Project1 - Setup (コード)
(General)                SetupApp
Private Sub SetupApp()
    Dim Ret As String
    Dim ExecFile As String

    ExecFile = Dir1.Path + "¥" + File1.FileName
    Ret = MsgBox("選択したアプリケーションを登録しますか？", _
                vbOKCancel, "アプリケーションの登録")
    If Ret = 1 Then
        i = GetSetting(appname:="QRun", section:="SetApp", _
                Key:="AppCount", Default:="0")
        i = i + 1

        SaveSetting appname:="QRun", section:="SetApp", _
            Key:=i, setting:=ExecFile

        main.List1.AddItem ExecFile
    End If

    SaveSetting appname:="QRun", section:="SetApp", _
        Key:="AppCount", setting:=i

    Setup.Hide
End Sub
```

ここをクリックする

ブレークポイントが設定された

3 プログラムの実行ボタンをクリックします。通常どおりプログラムは実行しますが、ブレークポイントのある行のところ（セットアッププログラムでアプリケーションを登録したとき）で、プログラムの動作が停止します。ステップ実行のときのように、ブレークポイントを設定した行が黄色の反転表示になります。

これで、このままF8キーを押せば、ステップインで1行ずつプログラムを実行できます。そのまま動作させたければ、ツールバーの［継続］ボタン▸をクリックすれば、通常の実行が行われます。

Project1 - Setup (ｺｰﾄﾞ)

(General) SetupApp

```
Private Sub SetupApp()
    Dim Ret As String
    Dim ExecFile As String

    ExecFile = Dir1.Path + "¥" + File1.FileName
    Ret = MsgBox("選択したアプリケーションを登録しますか？", _
            vbOKCancel, "アプリケーションの登録")
    If Ret = 1 Then
        i = GetSetting(appname:="QRun", section:="SetApp", _
                Key:="AppCount", Default:="0")
        i = i + 1

        SaveSetting appname:="QRun", section:="SetApp", _
            Key:=i, setting:=ExecFile

        main.List1.AddItem ExecFile
    End If

    SaveSetting appname:="QRun", section:="SetApp", _
        Key:="AppCount", setting:=i

    Setup.Hide
End Sub
```

ここでブレークする

4 設定したブレークポイントを解除するには、ブレークポイントの●印を再度クリックします。

また、ブレークポイントはいくつも設定できますので、確認したい場所だけステップイン実行させる、ということが可能になります。

COLUMN・・・・・

タイトルバーによるモードの確認

Visual Basicのタイトルバーには、Visual Basicの動作状態が表示されます。通常、プログラム作成中は、「デザイン」という文字が表示されます。

また、プログラムを実行すると「実行」が、ステップイン実行で動作を一時中止すると「中断」と表示され、現在Visual Basicがどの動作モードにあるのかがわかるようになっています。

まとめ

長いコードで、特定の場所だけステップ実行させたいときは、このブレークポイント機能を使うと便利です。

また、プログラムのテストでバグが出たときなどは、バグがありそうな場所にブレークポイントを設定しながらバグの個所を探し出す、という操作が可能になります。

ワンポイントアドバイス

イミディエイトウィンドウを使うと、プログラムのコードとは関係なく、自分の考えたコードを試すことができます。

下の画面は、組み込んだフォームを表示させるコードを、イミディエイトウィンドウに記述しています。コードを記述後、Enterキーを押すと、コードが実行されます。プログラムが実行されているわけではありません。

Form1

Command1

コードが実行される

イミディエイト

Form1.Show

コードを書いてEnterキーを押す

COLUMN

Debug Print

変数の内容を知るのに、ウォッチウィンドウを使う方法を説明しましたが、Debugオブジェクトのメソッドを使うと、イミディエイトウィンドウに変数の中身を表示できます。

例えば、変数「NumCount」の中身を表示するには、コードの中で、

```
Debug.print NumCount
```

と記述します。もちろん、デバッグ用のオブジェクトとメソッドなので、デバッグが終われば、このコードは削除します。

Lesson 34

第８日
３時限目

デバッグと
エラー処理

Lesson34.vbp
VB6CD
Lesson34

◎Learning Edition

エラー処理を使う

ユーザーが入力したファイル名から、ファイルのサイズを求めるプログラムを作成し、「OnError」ステートメントを使った、エラー処理の組み立て方をマスターします。

● 今回作成するプログラム

Form1
ファイル名をフルパスで入力してください。
C:¥form1.frm
サイズの取得

1 ファイル名を入力して…

2 ボタンをクリックする

Microsoft Visual Basic
実行時エラー '53':
ファイルが見つかりません。
継続(C) 終了(E) デバッグ(D) ヘルプ(H)

エラー処理をしないとプログラムが停止してしまう

Lesson34
指定したファイルが見つかりません。再度入力して下さい。
OK

エラー処理でメッセージボックスを表示しプログラムを継続

POINT

プログラムの処理の中で、ユーザーの操作によっては正常な動作ができなくなる場合があります。例えば、ファイル名を入力してもらう場面で、存在しないファイル名を入力されると、プログラムは正常な処理ができなくなります。このような場合は、プログラム側で存在しないファイル名が入力された場合の処理を、設けておく必要があります。これをエラー処理といいます。

Visual Basic では、メソッドなどではエラーが発生するとプログラムの動作を停止させてしまうものがあります。そこで、これを回避するために、「エラー処理」専用のステートメントが用意されています。

ここでは、このエラー処理ステートメントの使い方を紹介します。

コントロールとプロパティの設定

※新しいプロジェクトとフォームを用意します。

1 ツールボックスの［Label］ A をクリックし、フォーム上にラベルコントロールを配置する

Form1
Label1

1 ここをクリックして…
2 そのままここまでドラッグ

2 ［Caption］プロパティの値を「ファイル名をフルパスで入力して下さい」に変更する

プロパティ－Label1
Label1 Label
全体 項目別

BackColor	&H8000000F&
BackStyle	1 － 不透明
BorderStyle	0 － なし
Caption	ファイル名をフルパスで
DataField	
DataFormat	
DataMember	

1 ここをダブルクリック
2 値を入力

3 ツールボックスの［TextBox］ abl をクリックし、フォーム上にテキストボックスコントロールを配置する

Form1
ファイル名をフルパスで入力してください。
Text1

1 ここをクリックして…
2 そのままここまでドラッグ

ラベルを選択

この操作を行う前に、作成したラベルコントロールが選択されていることを確認してください。

テキストボックスを選択

この操作を行う前に、作成したテキストボックスコントロールが選択されていることを確認してください。

4 プロパティウィンドウで［Text］プロパティの値を削除する

プロパティ	値
TabIndex	2
TabStop	True
Tag	
Text	
ToolTipText	
Top	600

1 ここをダブルクリック

2 Delキーを押す

5 ツールボックスの［ComandButton］ をクリックし、フォーム上にコマンドボタンコントロールを配置する

Form1

ファイル名をフルパスで入力してください。

Command1

1 ここをクリックして…

2 そのままここまでドラッグ

コマンドボタンを選択

この操作を行う前に、作成したコマンドボタンコントロールが選択されていることを確認してください。

6 プロパティウィンドウで［Caption］プロパティの値を「サイズの取得」に変更する

プロパティ－Command1

Command1 CommandButton

全体 | 項目別

プロパティ	値
Appearance	1 - 3D
BackColor	&H8000000F&
Cancel	False
Caption	サイズの取得
CausesValidation	True
Default	False
DisabledPicture	(なし)

1 ここをダブルクリック

2 値を入力

7 ツールボックスの［Label］ A をクリックし、フォーム上にラベルコントロールを配置する

Form1

ファイル名をフルパスで入力してください。

サイズの取得

Label2

1 ここをクリックして…

2 そのままここまでドラッグ

8 プロパティウィンドウで［Caption］プロパティの値を削除する

プロパティ - Label2

Label2 Label

全体 | 項目別

BackColor	&H8000000F&
BackStyle	1 - 不透明
BorderStyle	0 - なし
Caption	
DataField	
DataFormat	

1 ここをダブルクリック

2 Delキーを押す

ラベルを選択

この操作を行う前に、作成したラベルコントロールが選択されていることを確認してください。

コードの記述

Form1:Command1-Click

Project1 - Form1 (コード)

Command1 | Click

```
Private Sub Command1_Click()
    Dim FName As String
    Dim FSize As Long

    FName = Text1.Text
    FSize = FileLen(FName)
    Label2.Caption = "ファイル" & FName & " のサイズは " _
                     & FSize & " バイトです。"
End Sub
```

解説

1 プログラムは、［サイズの取得］ボタンをクリックしたときに実行させるので、コマンドボタンのClickイベントプロシージャに作成します。

まず、ファイル名とファイルサイズを格納するための変数を宣言します。ファイルサイズの変数FSizeは、Long（長整数型）で宣言してください。

```
Dim FName As String
Dim FSize As Long
```

2 テキストボックスコントロールに入力されたファイル（パス）名は、変数「FName」に格納します。

そして、格納した変数を「FileLen」関数に渡します。「FileLen」関数は、指定したファイルのサイズを返す関数です。返してきたサイズは、変数で受け取って文字列と組み合わせて、もう1つのラベルコントロールで表示します。

```
FName = Text1.Text
FSize = FileLen(FName)
Label2.Caption = "ファイル " & FName & " のサイズは " _
                    & FSize & " バイトです。"
```

3 これで完成です。プログラムを動作させてください。ファイル名を入力すると、そのファイルサイズを表示します。

ただし、ファイル名をドライブ名を含めたフルパスで入力しないと、FileLen関数はエラーになって、プログラムを強制的に終了させてしまいます。また、存在しないファイル名を入力しても同様です。

Microsoft Visual Basic
実行時ｴﾗｰ '53':
ﾌｧｲﾙが見つかりません。
継続(C)　終了(E)　ﾃﾞﾊﾞｯｸﾞ(D)　ﾍﾙﾌﾟ(H)

4 この事態を回避するために、Visual Basicにはエラー処理のステートメントが用意されています。ここでは、その1つ「On Error」ステートメントを使います。

このステートメントを使うと、Visual Basicが発生させたエラーを受け取って、指定した処理にプログラムのフローをジャンプさせることができます。

このプログラムでは、誤ったファイル名の入力があった場合、入力間違いであることをメッセージボックスで通知し、再度入力してもらうように処理を戻します。

5

では、実際にこの処理を組み込んでみましょう。
「FSize = FileLen(FName)」のところでエラーが発生しますから、この行の前にエラー処理を次のように組み込みます。この「On Error」ステートメントでは、エラーが発生したらラベル「MSG」の処理にジャンプするように記述しています。

```
On Error GoTo MSG
Fsize = FileLen (FName)
```

6

エラーが発生しなかったときには、エラー処理へジャンプする必要はありませんから「Exit Sub」という行を、エラー処理ルーチンである「MSG」ラベルの前に記述しておきます。エラーが発生しなかったときは、この行でClickイベントプロシージャの処理が終了します。

```
Exit Sub
```

7

「GoTo」ステートメントは、その次に記述してある「ラベル」に処理をジャンプさせます。この処理では、エラーが発生すると「MSG」というラベルに処理をジャンプさせ、FileLen関数の次の処理は実行されません。
ラベルは、ラベル名の後に「:」を付けて、プロシージャの終わりのほうに設定します。

```
MSG:
```

8

このラベルにジャンプすると、ラベル以降に記述した処理が実行されます。そして、その処理が終了すると、Clickプロシージャの処理も終了します。
ここでは、エラーが発生すると、ファイルが見つからないというメッセージボックスを表示し、テキストボックスコントロールの［Text］プロパティを空白にし、フォーカスを設定して次の入力に備えています。

```
MsgBox "指定したファイルが見つかりません。再度入力して下さい。"
Text1.Text = ""
Text1.SetFocus
```

Lesson34

指定したファイルが見つかりません。再度入力して下さい。

OK

9 これで、エラー処理の完成です、最後にエラー処理を付け加えたコード全体を記載しておきます。

```
Private Sub Command1_Click()
    Dim FName As String
    Dim FSize As Long

    FName = Text1.Text
    On Error GoTo MSG           'エラー処理モードに入る
    FSize = FileLen(FName)
    Label2.Caption = "ファイル " & FName & " のサイズは " _
                & FSize & " バイトです。"
    Exit Sub        'エラーが発生しなかったときはここでプロシージャを終了

MSG:                'エラーが起こるとここにジャンプする
    MsgBox "指定したファイルが見つかりません。再度入力して下さい。"
    Text1.Text = ""
    Text1.SetFocus
End Sub
```

エラー処理ルーチンを組み込んだプログラムの流れ

まとめ

　エラー処理のステートメントは、「On Error」以外にもいくつか用意されています。また、メソッドや関数ではエラーが発生すると「エラーコード」を返すものもあります。この場合は、Select Caseステートメントなどを使って、エラーコードにあったエラー処理を組み立てることができます。
　エラー処理は、プログラムを安定して動作させるために必要な処理ですから、きちんとしたエラートラップを設けて、品質の高いプログラムを作成してください。

練習問題

Q1　このプログラムで、エラー発生時のエラー番号をメッセージボックスで表示してください。

解答は巻末に→

ワンポイントアドバイス

　Errオブジェクトの［Number］プロパティは、エラーが発生するとそのエラー番号を格納します。エラー番号によって処理を振り分けたい場合は、このプロパティを活用します。
　エラー番号とは、発生したエラーの種類を番号で表示する機能です。エラー番号がわかれば、どういう理由でエラーが発生したのかがわかります。
　例えば、エラー番号13は「型が一致しません」という内容のエラーです。番号とそのエラーの意味については、「エラー番号一覧」として付録に掲載しているので参考にしてください。

Let's VBSchool!!

第９日 (Lesson35～Lesson36)
実行ファイルの作成と配布

プログラムが最終的に完成したら「実行可能ファイル」（EXEファイル）にしましょう。このファイル形式で保存すれば、Visual Basicを起動せずに作成したプログラムを実行することが可能になります。

ただし、この実行可能ファイルを、Visual Basicがインストールされていない他のパソコンへ配布しても、そのままでは動きません。動作させるには、EXEファイル以外にも「ランタイムDLL」と呼ばれるダイナミンクリンクライブラリや、そのプログラムが使用している外部コンポーネントが必要になります。このような場合は、ディストリビューションウィザードを使えば、配布時にこれらの必要なファイルを自動的に収集してくれます。

今日のレッスンで解説する項目

- ❏ フォームにアイコンを割り当て、実行可能なEXEファイルの作成方法
- ❏ プログラムにバージョン番号を付加する方法
- ❏ プログラムのバージョン情報を組み込む方法
- ❏ ディストリビューションウィザードを使って、作成したプログラムを配布できるインストールキットにする方法

Lesson 35

第９日
１時限目

実行ファイルの作成と配布

Lesson35.vbp
VB6CD
Lesson35

◎Learning Edition

EXEファイルの作成

フォームにアイコンを割り当て、実行可能なEXEファイルの作成方法を覚えます。

今回作成するプログラム

D:¥_GENKOU¥10ka-VB6-1¥付録¥LESSON35

ファイル(F) 編集(E) 表示(V) 移動(G) お気に入り(A) ヘ

アドレス 0ka-VB6-1¥付録¥LESSON35

Lesson35

アイコンを選択すると、その説明が表示されます。

Club.ico
Diamond.ico
Earth2.ico
Heart.ico
Launch.exe
Lesson35.vbp
Lesson35.vbw
Main.frm
Main.frx
Mssccprj.scc
Setup.frm
Setup.frx
Spade.ico

13 個のオブジェクト　マイ コンピュータ

1 launch.exeファイルを実行すると…

プログラムランチャー

登録済みアプリケーション

c:¥WINNT¥SYSTEM32¥CALC.EXE
c:¥WINNT¥SYSTEM32¥cdplayer.exe
c:¥WINNT¥SYSTEM32¥notepad.exe

追加　削除

Run　終了

2 プログラムランチャーが起動する

POINT

Lesson25の練習問題で使用したプログラムランチャーを使って、単独で実行可能なEXEファイルを作成します。実行ファイルを作成することを「コンパイルする」ともいいます。

コントロールとプロパティの設定

1 「Q&A-25.vbp」プロジェクトファイルを開く

2 mainフォームを選択し、プロパティウィンドウで［Icon］プロパティをクリックして［…］をクリックする

Height	4500
HelpContextID	0
Icon	(ｱｲｺﾝ)
KeyPreview	False
Left	5115
LinkMode	0 - なし

1 ここをクリック
2 ここをクリック

3 「Heart.ico」アイコンを選択し、［開く］をクリックする

1 ここをクリック
2 ここをクリック

Q&A-25.vbp

このファイルは、付録CD-ROMの「Q&A」フォルダに収録されています。

Mainフォーム

mainフォームは［プログラムランチャー］フォームです。プロジェクトエクスプローラで、main（Q&A-251.frm）をダブルクリックすると、mainフォームを選択できます。

「Heart.ico」

このアイコンファイルは、付録CD-ROMの「Lesson35」フォルダに収録されています。

4 mainフォームのタイトルバーにハートのアイコンが表示される

5 同様に、Setupフォームの［Icon］プロパティに［Earth2.ico］ファイル（地球アイコン）を設定する

6 ［ファイル］メニューの［launch.exeの作成］をクリックする

7 ［OK］ボタンをクリックする

Setupフォーム

Setupフォームは［実行プログラムの登録］フォームです。プロジェクトウィンドウで、［Setup（Q&A-252.frm）］をダブルクリックすると、Setupフォームを選択できます。

実行ファイルの保存

実行（EXE）ファイルは、同じファイル名で何回もコンパイルできますが、その場合は上書き保存になります。

プログラムを修正するたびに、別々の実行ファイルにしたい場合は、ファイル名を少し変えるか、出力先のフォルダを変更します。

8 実行ファイル（Launch.exe）が作成される

Lesson35

名前	サイズ	種類
Club.ico	2KB	アイコン
Diamond.ico	2KB	アイコン
Earth2.ico	2KB	アイコン
Heart.ico	2KB	アイコン
Launch.exe	28KB	アプリケーション
Lesson35.vbp	1KB	Visual Basic Project
Main.frm	5KB	Visual Basic Form ...

アイコンを選択すると、その説明が表示されます。

作成された実行ファイル

解　説

1　単独で実行できるファイルを作成すると、Visual Basicを起動しなくても、作成したプログラムをフォルダから直接起動させることができます。

また、メインのフォームに組み込んだアイコンが、プログラムのアイコンになり、フォルダに表示されます。フォームのアイコンは、プログラムをタスクバーに格納したときにも表示されます。

2　EXEファイルを作成すると、他のコンピュータに作成したプログラムをインストールして、使うことができます。インストールするコンピュータにVisual Basicが組み込まれていない場合は、このあとに説明する「ディストリビューションウィザード」を使って、関連するファイルをいっしょにインストールする必要があります。

ワンポイントアドバイス

アイコンファイルとは、拡張子が「.ico」の、専用フォーマットで作成されたグラフィックファイルです。

Visual Basicのフォルダにある［samples］フォルダに、いくつかのアイコンファイルが入っていますが（オプションでインストール）、使用したいものがないときは自分で用意するしかありません。フリーウェアやシェアウェアでアイコンファイルがインターネット上にかなり流通しているので、これらを使うのも一つの手段です。また、アイコンを自作する場合は、専用のアイコンエディタが必要になりますが、これもインターネットが利用できる環境であれば、フリーウェアシェアウェアのサイトから入手できます。次のURLのホームページ「winfiles.com」には、たくさんのアイコンファイルやアイコンエディタが登録されているので、利用してはいかがでしょうか。

http://www.winfiles.com/apps/98/icons.html

なお、作成したプログラムを営利目的に使う場合は、くれぐれも著作者の許可を得て使うようにしてください。

Lesson 36(1)

第９日
２時限目

実行ファイルの作成と配布

Lesson36.vbp
VB6CD
Lesson36

◎Learning Edition

バージョンと作成情報を設定する

バージョン番号を付加する方法を覚えます。また、プログラムのバージョン情報を組み込む方法も紹介します。

今回作成するプログラム

Launch.exe ファイルのプロパティを表示すると…

Launch.exe ファイルのバージョン情報が表示される

POINT

引き続き、ランチャープログラムを使って、バージョン番号を組み込む方法と、著作権などの制作情報をプログラムに組み込む方法を紹介します。

コントロールとプロパティの設定

1 ［ファイル］メニューの［launch.exeの作成］をクリックする

1 ここをクリック

2 ここをクリック

2 ［オプション］ボタンをクリックする

1 ここをクリック

［プロジェクトプロパティ］ダイアログボックスが表示される

バージョン番号

作成したプログラムに、一連番号を付けることができます。これが、「バージョン番号」と呼ばれるもので、Visual Basicには自動的にこの番号を付加する機能が装備されています。

［コンパイル］タブ

［コンパイル］タブは、コンパイルの方法を選択するところです。特にデフォルトの設定で、変更する必要はないでしょう。

なお、このタブはLearning Editionでは表示されません。

自動インクリメント

プログラムをコンパイルするたびに、Visual Basicが自動的に［リビジョン］の番号を1つずつ増やしてくれる機能です。

3 ［バージョン番号］の［メジャー］を「1」のままにし、［自動インクリメント］をチェックする

Project1 - プロジェクト プロパティ
実行可能ファイルの作成 | コンパイル
バージョン番号
メジャー(M): 1 マイナ(I): 0 リビジョン(R): 0
☑ 自動インクリメント(U)
バージョン情報
種類(Y): 値(V):
アプリケーション
タイトル(T): Lesson36
アイコン(O): main

1 ここをクリック

4 ［バージョン情報］の［種類］ボックスで種類をクリックし、［値］ボックスに次の値を入力する

Project1 - プロジェクト プロパティ
バージョン情報
種類(Y): コメント / ファイルの説明 / 会社名
値(V): 初期バージョン。テスト後修正あり

1 種類をクリック
2 値を入力

種類	値
コメント	初期バージョン。テスト後修正あり
ファイルの説明	サンプルプログラム・ソフトウェアランチャー 登録できるプログラムに制限あり。
会社名	瀬戸 遥 & 翔泳社
製品名	ソフトウェアランチャー Launch.exe
著作権	瀬戸 遥 1998.11

5 入力できたら［OK］ボタンをクリックする

Project1 - プロジェクト プロパティ
種類(Y): 商標 / 製品名 / 著作権
値(V): 瀬戸遥 1998.11.
コマンド ライン引数(C):
条件付きコンパイル引数(D):
☑ 使用しない ActiveX コントロールについての情報を削除する(E)
OK キャンセル ヘルプ

1 ここをクリック

6 もう一度［OK］ボタンをクリックして、実行ファイルを作成する

実行可能ﾌｧｲﾙの作成
保存する場所(I): Lesson36
ﾌｧｲﾙ名(N): launch.exe
OK
ｷｬﾝｾﾙ
ﾍﾙﾌﾟ(H)
ｵﾌﾟｼｮﾝ(O)...

1 ここをクリック

7 ファイルを保存したフォルダを開き、「Launch.exe」のショートカットメニューから［プロパティ］を選ぶ

D:¥_GENKOU¥10ka-VB6-1¥付録¥LESSON36
ファイル(F)　編集(E)　表示(V)　移動(G)　お気に入り(A)　ヘルプ(H)
アドレス 0ka-VB6-1¥付録¥LESSON36
Lesson36
Setup
Earth2.ico
Heart.ico
Launch.exe
Launch.exe
ｱﾌﾟﾘｹｰｼｮﾝ
更新日時
98/09/14 午前
10:32
サイズ: 32KB
属性: アーカイブ
開く(O)
クイック ビューア(Q)
Add to Zip
送る(T)
切り取り(T)
コピー(C)
ショートカットの作成(S)
削除(D)
名前の変更(M)
プロパティ(R)
選択されたオブジェクトのプロパティを表示します。

1 ここを右クリック

2 ここをクリック

作成された実行ファイル

実行ファイルは、［実行可能ファイルの作成］ダイアログボックスの［保存する場所］ボックスで指定したフォルダの中に作成されます。

ショートカットメニューの表示

「Launch.exe」ファイルのアイコンをマウスの右ボタンでクリックしてください。

8 ［バージョン情報］タブをクリックして確認する

Launch.exeのプロパティ

全般　バージョン情報

ファイル バージョン: 1.00.0002

説明　サンプルプログラム・ソフトウェアランチャー 登録で

著作権　瀬戸遥 1998.11.

詳細

項目:　情報

コメント
会社名
言語
正式ファイル名
製品バージョン
製品名
内部名

初期バージョン。テスト後修正あり

OK　キャンセル　適用(A)

1 ここをクリック

2 項目をクリックすると…

3 それに対する情報が表示される

COLUMN ・・・・・

著作権、プログラム配布の注意

自分で作成したプログラムを、パソコン通信やインターネットで配布する場合は、著作権の問題を明確にしておく必要があります。

日本の法律では、プログラム（ソースコード）といえども自動的に著作権が発生します。そのため、プログラムの作者に無断で勝手にそのプログラムを販売したり配布すると、法律で罰せられます。

明らかに他人のソースコードをそのまま流用し再配布する場合には、必ずそのコードの作者に許可を得る必要があります。ただ、コードの「考え方」やプログラムの構造だけを拝借し、後は自分のやり方で1から組み上げたような場合には、許可の必要がない場合があります。しかし、このあたりの線引きもややケースバイケースな所がありますので、心配なときは必ず「著作権者への連絡」を怠らないようにしましょう。それが無用なトラブルを避ける最良の手段です。

プログラムを配布する際は、フリーソフトであっても、配布形態や営利目的での転載・掲載について、自分の立場を明確にするドキュメントなどを添付して配布するようにしましょう。

解説

1　［実行可能ファイルの作成］ダイアログボックスの、［オプション］をクリックして表示される、［プロジェクトプロパティ］ダイアログボックスで設定したバージョン番号とバージョン情報は、プログラムを作成する上での情報を記録するのに使います。

バージョン番号は、プログラムに修正を加えるたびに変更し、最新のプログラムには最新の番号が割り当てられるようにしておきます。

プログラム開発中は、常に修正されたり機能が変更になったりしますので、どのプログラムが何番目に作成したものなのかを、このバージョン番号で把握するようにします。

バージョン番号は、［メジャー］→［マイナー］→［リビジョン］の順に小さくなっていきます。ちょっとした修正では、［リビジョン］の数字を増加させます。このあたりの数字の使い方は、自分で決めてかまいません。

2　［自動インクリメント］にチェックを付けると、コンパイルするたびに、Visual Basicが自動的に［リビジョン］の番号を1つずつ増やしてくれます。

［バージョン情報］は、プログラムの著作権や会社名、補足説明などを記録しておきます。自分がプログラムの情報を把握しておくのに使いますが、ユーザーにプログラムの情報を伝えるためにも役立ちます。

これらの情報は、フォルダからプログラムのプロパティを使ってユーザーが自由に見ることができます。

ワンポイントアドバイス

設定したバージョン番号およびバージョン情報は、Appオブジェクトのプロパティを使って、プログラムで使うことができます。フォームモジュールテンプレートの「バージョン情報」も、これらのプロパティを使用しています。

<バージョン番号のプロパティ>

プロパティ名	説明
Major	メジャー
Minor	マイナー
Revision	リビジョン

<バージョン情報のプロパティ>

プロパティ名	説明
Comments	コメント
CompanyName	会社名
LegalCopyright	著作権
ProductName	製品名

Lesson 36(2)

第９日
３時限目

実行ファイルの作成と配布

Lesson36.vbp
VB6CD
Lesson36

◎Learning Edition

ディストリビューションウィザード

ディストリビューションウィザードを使って、作成したプログラムを配布できるインストールキットにする方法を覚えます。ここでは、「Lesson36.vbp」の実行ファイル「Launch.exe」の配布キットを、フロッピーディスクを使って配布する場合を例にとって説明します。

POINT

作成したプログラムを、Visual Basicが組み込まれていない他のコンピュータで動作させる場合は、使用したコントロールファイルや「ランタイムDLL」と呼ばれる付属ファイルがないと、正常に動作しません。

Visual Basicに付属する「ディストリビューションウィザード」は、作成したプログラムファイルを配布するためのキットを、自動作成してくれる便利なウィザードです。

ここでは、このディストリビューションウィザードの使い方を説明します。

コントロールとプロパティの設定

1 ［スタート］メニューから［ディストリビューションウィザード］を選ぶ

1 ここをクリック

2 ［参照］ボタンをクリックし、プロジェクトファイル（Lesson36.vbp）を指定する

1 ここをクリック

2 ファイルを指定

3 ここをクリック

ここで説明する手順は...

ここでは、お使いのコンピュータにVisual Basic 6.0をインストールしている場合の手順を説明します。

このウィザードを起動するには

Windows95のタスクバーにある［スタート］ボタン→［プログラム］→［Microsoft Visual Basic 6.0］→［Microsoft Visual Basic 6.0 Tools］→［ディストリビューションウィザード］の順にクリックします。

プロジェクトファイルを指定する

ディストリビューションウィザードでは指定したファイルがプロジェクトファイルでも、自動的に実行ファイルに再コンパイルしてから配布キットに含めます。

ファイルの場所

［ファイルの場所］ボックスで、ファイルが保存されているフォルダを指定します。すると、その下のボックスにフォルダに入っているファイルの一覧が表示されます。

3 [パッケージ] をクリックする

1 ここをクリック

4 [標準セットアップパッケージ] をクリックし、[次へ] をクリックする

1 ここをクリック

2 ここをクリック

5 作業用のフォルダを選択し、[次へ] をクリックする

1 ドライブを指定

2 フォルダを指定

3 ここをクリック

コンパイルしていないときは？

プロジェクトファイルをコンパイルしていないときは、コンパイルをするかどうかのダイアログが表示されます。[コンパイル] をクリックして、最新の実行ファイルを作成しておきます。

2回目からは？

パッケージ作成の2回目以降は、前回作成されたスクリプトを選択する画面が、表示されます。

作業用フォルダ

事前に作業用のフォルダを作成しておきましょう。[新しいフォルダ] ボタンをクリックしてフォルダを作成することができます。「Package」などのわかりやすい名前を付けておくといいでしょう。

6 すべてのボックスがチェックされていることを確認し、[次へ] をクリックする

ディストリビューション ウィザード - 含まれるファイル

以下のリストのファイルがこのパッケージに含まれます。これ以外のファイルを追加する場合は [追加] をクリックしてください。ファイルをパッケージから削除する場合はファイル名の左のチェック ボックスをオフにしてください。

1 すべてチェックされていることを確認

2 ここをクリック

7 [複数の Cab ファイル]、[1.44MB] を選び、[次へ] をクリックする

ディストリビューション ウィザード - Cab ファイルのオプション

パッケージは、1 つのキャビネット ファイル (cab) にまとめるか、複数のキャビネット ファイルに分けて作成することができます。フロッピー ディスクで配布する場合は、複数のキャビネット ファイルに分けて、1 つの .cab ファイルのサイズをフロッピー ディスクのサイズ以下に指定する必要があります。以下で適切なオプションを選択してください。

1 ここをクリック

2 ここをクリック

3 ここをクリック

1.44MBで保存

この設定は2HD、1.44MBのフロッピーディスクで配部する場合の設定です。

ただし、フロッピーの状態によっては1.44MBだとコピーできない場合があります。そのときは、[1.2MB] を選択してcabファイルを作りなおしましょう。

8 インストール時のタイトル「Lesson36」を入力し、[次へ] をクリックする

ディストリビューション ウィザード - インストール時のタイトル

セットアップ プログラムの実行時に表示されるタイトルを入力してください。

1 タイトルを入力

2 ここをクリック

タイトルは自由

インストール時のタイトルは自由に設定してかまいません。

一般的にはプログラムの名前を付けます。

登録メニュー名の変更

登録するメニューの名前を変更したい場合は、P.358を参照してください。

セットアップ先の変更

セットアップ先を変更したい場合は、P.359を参照してください。

9 スタートメニューのメニュー名を確認し、［次へ］をクリックする

ディストリビューション ウィザード － [スタート] メニュー項目

インストール時に作成される [スタート] メニューのグループと項目を指定します。

[スタート] メニュー項目(S):
スタート メニュー
プログラム
Lesson36
Lesson36

新しいグループ(G)
新しい項目(I)...
プロパティ(P)...
削除(R)

ヘルプ　キャンセル　< 戻る(B)　次へ(N) >　完了(F)

1 メニュー名を確認

2 ここをクリック

10 セットアップ先を確認し、［次へ］をクリックする

ディストリビューション ウィザード － セットアップ先

このリスト内で各ファイルに割り当てられているマクロを変更してファイルのセットアップ先を変更できます。
$(ProgramFiles)¥MySubFolder のように、サブフォルダ情報をマクロの後に追加することもできます。

変更するファイルを選択して、[セットアップ先] 列の情報を変更してください。

ファイル(I):

名前	ソース	セットアップ先
launch.exe	D:¥arcwork¥Lesson36	$(AppPath)
VB6JP.DLL	C:¥WINNT¥SYSTEM32	$(WinSysPath)

ヘルプ　キャンセル　< 戻る(B)　次へ(N) >　完了(F)

1 セットアップ先を確認

2 ここをクリック

11 チェックを付けずに［次へ］をクリックする

ディストリビューション ウィザード － 共有ファイル

このパッケージのインストール時に、以下のファイルは共有ファイルとしてインストールできます。共有ファイルは複数のプログラムで使用できます。共有ファイルは、そのファイルを使用するプログラムがすべて削除された場合にのみ削除されます。共有ファイルとしてセットアップするファイルのチェック ボックスをオンにしてください。

共有ファイル(S):

名前	ソース	セットアップ先
☐ launch.exe	D:¥arcwork¥Lesson36	$(AppPath)

ヘルプ　キャンセル　< 戻る(B)　次へ(N) >　完了(F)

チェックしない

1 ここをクリック

12 ［完了］をクリックする

ディストリビューション ウィザード － 完了!

このパッケージの作成に必要な情報の収集が完了しました。このセッションの設定を保存する名前を入力してください。[完了] をクリックするとパッケージが作成されます。

スクリプト名(S):
標準のセットアップ パッケージ 1

ヘルプ　キャンセル　< 戻る(B)　次へ(N) >　完了(F)

1 ここをクリック

Lesson36.vbp － パッケージ レポート

[ウィザード]はこのアプリケーションの 2 キャビネット ファイルを作成しました。
キャビネット ファイルは 'j:¥work' に作成されました。

サポート ディレクトリ (j:¥work¥Support¥launch.BAT) にバッチ ファイルが作成されました。このファイルを使用するといくつかのファイルに変更を加えた場合にキャビネット ファイルを再作成できます。

レポートの保存(S)　閉じる

作成レポートが表示される

2 ここをクリック

J:¥work

ファイル(F)　編集(E)　表示(V)　移動(G)　お気に入り(A)　ヘルプ(H)

アドレス J:¥work

work

アイコンを選択すると、その説明が表示されます。

Support　setup.exe　launch1.CAB　launch2.CAB　Setup.lst

マイ コンピュータ

作成された「Launch.exe」のパッケージ

ウィザードの指定をやり直す

ウィザードの指定をやり直したい場合は、［戻る］ボタンをクリックすると前の画面に戻ることができます。

パッケージ

「パッケージ」とは、配布キットの元になる圧縮されたファイルやセットアッププログラム一式のことです。

13 [配置] ボタンをクリックする

GENKOU¥10ka-VB6-1¥付録¥LESSON36¥Lesson36.vbp 参照(B)...

パッケージ(P) このプロジェクトを、インターネット用 Cab ファイルや自動解凍セットアッププログラムなどの配布用の形式にパッケージ化します。

配置(D) このプロジェクトのパッケージを、インターネット サーバーなどの配布サイトに配置します。

スクリプトの管理(M) このプロジェクトのパッケージ スクリプトと配置スクリプトの名前の変更、コピー、および削除を行います。

閉じる ヘルプ

1 ここをクリック

14 配置スクリプトを確認し、[次へ] をクリックする

ディストリビューション ウィザード - 配置スクリプト

以下のリストから配置スクリプトを選択してください。

配置スクリプト(D): フロッピー配置 1

ヘルプ キャンセル < 戻る(B) 次へ(N) > 完了(F)

1 配置スクリプトを確認

2 ここをクリック

配置スクリプトが表示されない

配置スクリプトがまだ存在しないときは、このダイアログは表示されません。手順15に進んでください。

15 パッケージ名を確認し、[次へ] をクリックする

ディストリビューション ウィザード - 配置するパッケージ

配置するパッケージを選択してください。

配置するパッケージ(P): 標準のセットアップ パッケージ 1

ヘルプ キャンセル < 戻る(B) 次へ(N) > 完了(F)

1 パッケージ名を確認

2 ここをクリック

16 ［フロッピィディスク］が選択されていることを確認し、［次へ］をクリックする

ディストリビューション ウィザード - 配置方法

配置方法を選択してください。

配置方法(M):
フロッピー ディスク
フォルダ
Web に配置

説明(D):
複数のフロッピー ディスクにパッケージを配置するときに使用します。

ヘルプ　キャンセル　< 戻る(B)　次へ(N) >　完了(F)

1 ［フロッピーディスク］を確認

2 ここをクリック

17 フロッピィドライブを指定し、［次へ］をクリックする

ディストリビューション ウィザード - フロッピー ドライブ

以下のリストからフロッピー ドライブを選択してください。

フロッピー ドライブ(D):
A:¥

□ コピーする前にフォーマット(M)

ヘルプ　キャンセル　< 戻る(B)　次へ(N) >　完了(F)

1 ドライブを指定

2 ここをクリック

18 ［完了］ボタンをクリックする

ディストリビューション ウィザード - 完了!

ウィザードはこのパッケージを配置するのに必要な情報の収集を完了しました。このセッションの設定を保存する名前を入力してください。[完了] をクリックするとパッケージを配置します。

スクリプト名(S):
フロッピー配置 1

ヘルプ　キャンセル　< 戻る(B)　次へ(N) >　完了(F)

1 ここをクリック

フロッピードライブ

使用しているコンピュータのフロッピードライブのドライブ番号と使用するフロッピーディスクディスクのサイズに合わせて、それぞれ指定してください。

フロッピーに入り切らないときは？

もしフロッピーに入りきらない場合は、次のフロッピーを入れるようにウィザードが要求してきます。ディスクには分割された形で収録されます。

19 指定したドライブにフロッピーディスクをセットし、[OK] をクリックする

ディストリビューション ウィザード

セットアップ フロッピー ディスクの作成を開始します。ディスク 2 の 1 を挿入してください。

OK キャンセル

1 ここをクリック

コピーしています...

launch1.CAB
'work' から 'A:¥' へ

残り 35 秒

キャンセル

コピーが開始される

20 レポートが表示される

Lesson36.vbp - 配置レポート

j:¥work¥setup.exe をディスク 1 にコピーしました。
j:¥work¥SETUP.LST をディスク 1 にコピーしました。
j:¥work¥launch1.CAB をディスク 1 にコピーしました。
j:¥work¥launch2.CAB をディスク 2 にコピーしました。

レポートの保存(S) 閉じる

1 ここをクリック

21 作成されたファイル

A:¥

ファイル(F) 編集(E) 表示(V) 移動(G) お気に入

アドレス A:¥

3.5 インチ FD (A:)

launch1.CAB setup.exe Setup.lst

ディスク 1

3 個のオブジェクト マイ コンピ

A:¥

ファイル(F) 編集(E) 表示(V) 移動(G) お気に入

アドレス A:¥

3.5 インチ FD (A:)

launch2.CAB

ディスク 2

マイ コンピュータ

解説

1 　ディストリビューションウィザードを使うと、作成したプログラムを配布するために必要なファイルを、自動的に解析してキットとしてまとめてくれます。そして「圧縮」という技法を使ってファイルサイズを小さくし、配布しやすくしてくれます。

　このファイルは、Windows98のインストールで使われている、「Cab」ファイル（キャビネットファイル）という形式のファイルとして作成されます。

　また、ディストリビューションウィザードは、プログラムをコンピュータにインストールするための「セットアッププログラム」を、配布キットに組み込みます。このセットアッププログラムは、Visual Basicをコンピュータにインストールしたときに使ったものと同じように、プログラムをインストールするフォルダの指定などが行えます。

　配布キットに含まれているコントロール用のDLLファイルや、Visual BasicのランタイムDLLファイルは、それぞれWindowsのフォルダにインストールする必要がありますが、セットアッププログラムはこれらを自動で行ってくれます。

　Visual Basicで作成したプログラムを配布する場合は、このディストリビューションウィザードを使うことをお勧めします。

「Launch.exe」用のセットアッププログラムが起動

インストール先の指定も行える

インストール後はスタートメニューに組み込まれる

2 ディストリビューションウィザードで作成できる配布キットは、2種類あります。
1つは、今作成したように、プログラムを他のコンピュータにインストールするための配布キットです。もう1つは、インターネットのWWW・FTPサーバからダウンロードできる形態の配布キットです。いずれも、最初は「パッケージ」ボタンで、配布ファイルをパッケージにしてから、配布キットを作成します。

3 この配布キットでは、インストールするプログラム専用に、スタートメニューを作成してくれます。初期設定では、スタートメニューの［プログラム］メニューの中に、プログラムと同名のメニュー項目を作成します。
このメニュー名を変更するには、メニュー項目をクリックしてから［プロパティ］ボタンをクリックします。そして、新しいメニュー名を設定します。

4 インストールするプログラムとランタイムDLLのインストール先を変更したい場合は、「セットアップ先」画面で、「セットアップ先」をリストボックスから選択するか、任意のフォルダ名を入力します。

ディストリビューション ウィザード - セットアップ先

このリスト内で各ファイルに割り当てられているマクロを変更してファイルのセットアップ先を変更できます。
$(ProgramFiles)¥MySubFolder のように、サブフォルダ情報をマクロの後に追加することもできます。

変更するファイルを選択して、[セットアップ先] 列の情報を変更してください。

ファイル(I):

名前	ソース	セットアップ先
launch.exe	D:¥arcwork¥Lesson36	$(AppPath)
VB6JP.DLL	C:¥WINNT¥SYSTEM32	

$(AppPath)
$(ProgramFiles)
$(CommonFiles)
$(CommonFilesSys)
$(WinPath)
$(WinSysPath)
$(MSDAOPath)
$(Font)

ヘルプ　キャンセル　< 戻る(B)　次へ(N) >　完了(F)

セットアップ先を指定

まとめ

ディストリビューションウィザードを使うと、インストール先のコンピュータへのプログラム登録まで行ってくれる、配布キットの作成が行えます。

Windowsにスタートメニューとレジストリが採用されてから、これらの作業は簡単に行えないようになっていますので、このディストリビューションウィザードをぜひ活用してください。

このセットアッププログラムでインストールされたプログラムは、コントロールパネルの「アプリケーションの追加と削除」から削除します。

アプリケーションの追加と削除のプロパティ

インストールと削除 | Windows ファイル | 起動ディスク

フロッピー ディスクまたは CD-ROM から新しいアプリケーションをインストールするには、[インストール] をクリックしてください。

インストール(I)...

次のソフトウェアは自動的に削除できます。アプリケーションを削除したり、その構成ファイルを変更するには、一覧から選んで [追加と削除] をクリックしてください(T)

ACDSee 32
Adobe Acrobat Reader 3.0J
Adobe PhotoDeluxe for ファミリー 3.0
Adobe Photoshop 5.0J
DirectX ドライバ
JUSTSYSTEMアプリケーションの追加と削除
Lesson36
Microsoft IntelliPoint
Microsoft Office 97, Professional Edition
Microsoft Visual Studio 6.0 Enterprise Edition (日本語)

追加と削除(R)...

適用(A)

1 削除するアプリケーションをクリック

アプリケーションの削除

Lesson36 とそのすべてのコンポーネントを削除しますか?

はい(Y)　いいえ(N)

2 ここをクリック

アプリケーションの削除

削除されました。

OK

3 ここをクリック

ワンポイントアドバイス

Visual Basic のランタイム DLL、「VB6JP.dll」や「MSVBVM60.dll」は、Visual Basic がインストールされていないコンピュータで、Visual Basic で作成したプログラムを動作させるために必要なファイルですが、ファイルサイズがとても大きく、この2つで「1.5MB」もの容量があります。そのため、セットアッププログラムと合わせ、配布キットのサイズが圧縮状態でも 1.5MB になってしまいます。

一度これらのファイルがインストールされたコンピュータに、作成したプログラムをインストールする場合は、次からは「VB6JP.dll」「MSVBM60.dll」の2つのファイルは、配布キットに入れる必要はありませんので、配布キットのチェックをはずして、ファイルサイズを小さくできます。

ディストリビューション ウィザード － 含まれるファイル
以下のリストのファイルがこのパッケージに含まれます。これ以外のファイルを追加する場合は[追加]をクリックしてください。ファイルをパッケージから削除する場合はファイル名の左のチェック ボックスをオフにしてください。
ファイル(I):
名前 | ソース
SETUP1.EXE | I:¥Program Files¥VB98¥
ST6UNST.EXE | I:¥Program Files¥VB98¥
VB6 ランタイム および OLE オートメーション
VB6JP.DLL | C:¥WINNT¥SYSTEM32
VB6STKIT.DLL | C:¥WINNT¥SYSTEM32
追加(A)...
ヘルプ　キャンセル　< 戻る(B)　次へ(N) >　完了(F)
ここで配布キットの内容を調節する

V i s u a l B a s i c 6

Let's VBSchool!!

第10日 (Lesson37～Lesson38)
ActiveX コントロールを作る

L e a r n i n g S c h o o l

Visual Basic6.0 では、ツールパレットにあるコントロールを組み合わせて、新しいカスタムコントロールを作成できます。これらは「ActiveX コントロール」という新しい機能のコントロールとして他のプログラムから利用できます。

このレッスンでは、いくつかのコントロールを使って、独自の ActiveX コントロールを作成し、インターネットのホームページに組み込んで使う方法を紹介します。

なお、Learning Edition ではすでに登録済みの ActiveX コントロールを組み込むことはできますが、ActiveX コントロール自体を作成・登録する機能はありません。この日のレッスンは Professional Edition を使用して下さい。

今日のレッスンで解説する項目

❏ ActiveX コントロールの作成方法
❏ ActiveX コントロールの動作確認の方法
❏ ActiveX コントロールを生成する方法
❏ ActiveX コントロールの HTML への組み込み方法
❏ ActiveX コントロールを HTML ファイルとともに Web で公開する方法

ActiveX コントロールの概要

ActiveX コントロールを作成するにあたって、どのような特徴があるのか把握しておきましょう。

1

「ActiveX」は、Microsoft社が開発した新しいテクノロジーです。このテクノロジーを使ったコントロールが、「ActiveXコントロール」です。「ActiveXコントロール」は、従来「OLEコントロール」と呼ばれていたコントロールを改良し、いろいろな場面で利用できるように作られています。

これまでのVisual Basic用のコントロールは、Visual Basicでしか利用できませんでした。しかし、ActiveXコントロールは、「ActiveX」に対応しているアプリケーションであれば、どれでも利用できます。

現在では、Visual Basic以外に、「Visual C++」「Microsoft Office97」「インターネットエクスプローラ（Ver3.0以降）」などがActiveXコントロールを使える環境にあります。たとえば、「タブ付きダイアログボックスコントロール」は、Visual Basicのコントロールですが、Excel97のマクロ言語である「Visual Basic for Application」からも利用でき、また、「ActiveXコントロールパッド」や「FrontPage98」を使って、HTML文書にこのコントロールを組み込み、インターネットエクスプローラから利用することもできます。

このように、ActiveXに対応しているアプリケーションや環境なら、1つのコントロールをそれぞれで使用できるのが、ActiveXコントロールの特徴です。

Visual Basic で表示されている

Excel97（VBA）で表示されている

HTML に組み込まれている

2 これまで、Visual Basicでプログラムを作成するときは、新しいプロジェクトを選ぶときに「標準EXE」を選んでいました。ActiveXコントロールを作成するときは、この画面から「ActiveXコントロール」を選びます（P.365参照）。あとは、これまで行ってきたプログラム作成と同様の手法で、コントロールの配置やプロパティの設定でユーザーインターフェイスを作成し、メソッドとステートメントなどで処理を記述します。また、プログラムでは単独で実行可能なEXEファイルを生成しましたが、ActiveXコントロールではコントロールファイル（OCXファイル）を作成します。これらは、Visual Basicのツールボックスにある各コントロール類と、同じ種類のファイルです。

ActiveXコントロールを作成する場合は、「UserControl」オブジェクトに、既存のコントロールを組み合わせて作成しますが、標準EXEプログラムの作成に比べていくつかの制限があります。以下は、筆者が調べた限りの制限事項です。詳しくは、オンラインヘルプなどを参照してください。

①コントロール以外のオブジェクトをUserControlオブジェクトに配置することはできません。ツールボックスを右クリックし、ショートカットメニューから［コンポーネント］を選ぶと表示される［コンポーネント］ダイアログボックスの、［挿入可能なオブジェクト］タブに一覧表示されているオブジェクトは利用できません。
②UserControlオブジェクトには、メニューコントロールを組み込めません。
③HTMLから利用するActiveXコントロールでは、フォームを表示できません。
④OLEコンテナコントロールをUserControlオブジェクトで使用できません。

3 作成したActiveXコントロールは、ActiveXに対応したアプリケーションで、コントロールを利用できる環境であれば、どれでも使用できます。

Visual Basicでは、作成したActiveXコントロールをツールパレットに追加し、別のプログラムで再利用できます。Excel97やWord97など、Office97のアプリケーションでは、それぞれのマクロ言語（VBA）の中から利用できます。FrontPage98やFrontPage Expressを使うと、HTMLドキュメント内にActiveXコントロールを埋め込み、インターネットエクスプローラで操作できます。

Visual Basicで生成されたOCXファイルは、これらのアプリケーションの、ActiveXコントロールを組み込む機能の中から、自動的に呼び出すことができます。また、他のコンピュータやWebサーバに配置して利用することができます。

Visual Basicで作成した標準EXEプログラムは、独立したプログラムになっているため、他のアプリケーションなどで利用することができません。一方、ActiveXコントロールは、コントロールという形態なので、他の開発環境で自由に利用できるという、可搬性と再利用性を持っています。

プログラムを「部品」で作成しておけば、実現したい機能に合わせて部品を組み合わせていくことが可能になるので、プログラム開発やWebコンテンツの作成が効率的になります。

Lesson 37 (1)

第10日 1時限目

ActiveX コントロールを作る

Lesson37.vbp
VB6CD
Lesson37

×Learning Edition

スロットマシンコントロールの作成

ActiveX コントロールの作成方法を習得します。

今回作成するプログラム

SlotGame - Slot (UserControl)

Stop

アイコンファイルを4つ重ねる

ボタンを追加する

POINT

簡単なスロットゲームを、ActiveX コントロールとして作成します。実際に ActiveX コントロールを作成しながら、コントロールの作成方法を勉強します。

このコントロールは、タイマーコントロールで、4つのトランプのマークを交互に表示させせます。コマンドボタンをクリックすると、表示が停止します。

コントロールとプロパティの設定

1 [ファイル] メニューの [新しいプロジェクト] を選択し、[ActiveX コントロール] を選択する

1 ここをクリック

2 [開く] をクリック

2 プロパティウィンドウで [(オブジェクト名)] の値を「Slot」に変更する

1 ここをダブルクリック

2 「Slot」と入力

3 プロジェクトウィンドウでプロジェクトを選択し、[(プロジェクト名)] プロパティを「SlotGame」に変更する

1 プロジェクトを選択

2 ここをダブルクリック

3 「SlotGame」と入力

Learning 不可

Learning Edition ではこの表示画面はでません。

ActiveXの保存名

ActiveXコントロールでは、「Form1.frm」というフォームが保存される代わりに、「xxx.ctl」というUserControlがプロジェクトファイルといっしょに保存されます。

プロジェクトの保存

ActiveXコントロールを保存した後、必要があればプロジェクトファイルを保存してください。[保存する場所] ボックスで、フォルダを指定してください。

4 [標準] ツールバーの [上書き保存] ボタンをクリックし、名前を付けてプロジェクトを保存する

5 ツールボックスの [Image] をクリックし、フォーム上にイメージコントロールを1つ配置する

6 [標準] ツールバーの [コピー] ボタンをクリックし、続けて [貼り付け] ボタンをクリックする

7 メッセージに対して［はい］をクリックし、コントロール配列にする

Microsoft Visual Basic

既に同じ名前のコントロール 'Image1' があります。コントロール配列にしますか?

はい(Y) いいえ(N) ヘルプ

8 ［貼り付け］ボタンを2回クリックしてイメージコントロールをあと2つ作成し、それらをすべて重ねる

SlotGame - Slot (UserControl)

4つ重ねたイメージコントロール

9 プロパティウィンドウで［Image1(0) Image］を選択する

プロパティ - Image1(0)

Image1(0) Image
Image1(0) Image
Image1(1) Image
Image1(2) Image
Image1(3) Image
Slot UserControl
BorderStyle 0 - なし
DataField
DataFormat
DataMember

1 ここをクリック

2 ここをクリック

コントロール配列

コントロールをコピーすると、「コントロール配列」になります。コントロール配列では、最初の（コピー元の）コントロール名に（）を付けて配列の要素（番号）を加えたものが、コントロール配列のコントロール名になります。

配列番号がコントロール名に使われるので、変数でコントロールにアクセスすることができます。

同じ位置にオブジェクトを重ねるには

同じ位置にオブジェクトを正確に重ねるには、マウスで移動させるよりも［書式］メニューを利用すると便利です。

重ねたいオブジェクトを選択し、［書式］メニューの［整列］から2番目の［中央］を選択し、中央に揃えた後、再度［書式］メニューの［整列］から今度は5番目の［中央］を選択して、上下の同じ位置に揃えます。これでぴったりと重なります。

10 プロパティウィンドウで［Picture］プロパティに「Misc34.ico」ファイルを設定する

1 ここをクリック

2 ここをクリック

3 このファイルを選択

4 ［開く］をクリック

Misc34.ico

このファイルは、付録CD-ROMの「Lesson37」フォルダに収録されています。

11 ［Visible］プロパティの値を［True］に変更する

1 ［True］にする

12 手順1～12と同様の操作を行い、各イメージコントロールに次の値を設定する

コントロール名	[Picture]	[Visible]
Image1(1)	Misc35.ico	False
Image1(2)	Misc36.ico	False
Image1(3)	Misc37.ico	False

13 ツールボックスの［CommandButton］をクリックし、フォーム上にコマンドボタンを配置する

SlotGame - Slot (UserControl)

Command1

1 ここをクリックして…

2 そのままここまでドラッグ

14 プロパティウィンドウで［Caption］プロパティの値を「Stop」に変更する

プロパティ - Command1

Command1 CommandButton

全体 | 項目別

(オブジェクト名)	Command1
Appearance	1 - 3D
BackColor	&H8000000F&
Cancel	False
Caption	Stop
CausesValidation	True
Default	False
DisabledPicture	(なし)

1 ここをダブルクリック

2 「Stop」と入力

コマンドボタンを選択

この操作を行う前に、作成したコマンドボタンコントロールが選択されていることを確認してください。

15 ツールボックスの［Timer］をクリックし、フォーム上にタイマーコントロールを配置する

SlotGame - Slot (UserControl)

1 ここをクリックして…

2 そのままここまでドラッグ

16 プロパティウィンドウで［Interval］プロパティの値を「100」に変更する

プロパティ - Timer1

Timer1 Timer

全体 | 項目別

(オブジェクト名)	Timer1
Enabled	True
Index	
Interval	100
Left	3780

1 ここをダブルクリック

2 「100」と入力

タイマーを選択

この操作を行う前に、作成したタイマーコントロールが選択されていることを確認してください。

コードの記述

Slot:(General)-(Declarations)

SlotGame - Slot (コード)

(General) (Declarations)

```
Dim Flag As Integer
Dim Mode As Boolean
```

Slot:Timer1-Timer

SlotGame - Slot (コード)

Timer1 Timer

```
    If Flag = 0 Then
        Image1(0).Visible = True
        Image1(1).Visible = False
        Image1(2).Visible = False
        Image1(3).Visible = False
        Flag = 1
    ElseIf Flag = 1 Then
        Image1(0).Visible = False
        Image1(1).Visible = True
```

```
        Image1(2).Visible = False
        Image1(3).Visible = False
        Flag = 2
    ElseIf Flag = 2 Then
        Image1(0).Visible = False
        Image1(1).Visible = False
        Image1(2).Visible = True
        Image1(3).Visible = False
        Flag = 3
    ElseIf Flag = 3 Then
        Image1(0).Visible = False
        Image1(1).Visible = False
        Image1(2).Visible = False
        Image1(3).Visible = True
        Flag = 0
    End If

End Sub
```

Slot:Command1-Click

SlotGame - Slot (コード)

Command1 | Click

```
Private Sub Command1_Click()
    If Mode = True Then
        Timer1.Interval = 0
        Mode = False
        Command1.Caption = "Start"
    Else
        Timer1.Interval = 100
        Mode = True
        Command1.Caption = "Stop"
    End If
End Sub
```

Slot:UserControl-Initialize

SlotGame - Slot (コード)

UserControl | Initialize

```
Private Sub UserControl_Initialize()
    Flag = 1
    Mode = True
End Sub
```

解説

※UserControlのコードウィンドウを表示するには、フォームと同様にマウスでコントロールやオブジェクトをダブルクリックするか、プロジェクトウィンドウの［コードの表示］ボタンをクリックします。

1

このプログラムは、すでにアニメーションプログラム（Lesson18）で使用したテクニックの応用です。タイマーコントロールのTimerイベントが発生するたびに、4つ配置したイメージコントロールのVisibleプロパティを切り替え、表示するイメージを切り替えていきます。

2

コードウィンドウの宣言セクション（General）に、変数Flagを宣言します。この変数は、現在表示しているイメージコントロールの状態を把握するのに使用します。

また、Modeという変数も宣言します。この変数は、タイマーコントロールの動作状態を把握するのに使用し、データ型は論理型（Boolean）で宣言します。

```
Dim Flag As Integer
Dim Mode As Boolean
```

3

タイマーコントロールのTimerイベントプロシージャに、イメージコントロールの表示切り替え処理を記述します。変数Flagの値を調べ、イメージコントロールのVisibleプロパティの値を切り替えます。Image1(0)だけ表示するには、Image1(0)以外のイメージコントロールのVisibleプロパティをFalseに設定します。そして、次の表示にそなえ、変数Flagの値を「1」に設定します。

この処理を4つ設定すれば、Timerイベントが発生する毎に、順番にイメージが表示されます。

4

ActiveXコントロールが起動したときは、イメージコントロールのスロットは動作している状態（Mode = True）なので、最初は「Stop」の処理を行います。

タイマーコントロールの動作を決めるのは、［Interval］プロパティです。この値を「0」にすると、スロットの動作は停止します。

```
If Mode = True Then
    Timer1.Interval = 0
```

次に現在「True」に設定されている変数Modeに動作が停止状態にあることを示す「False」を代入します。

```
Mode = False
```

最後に次のボタンの動作がわかるように、コマンドボタンの［Caption］プロパティを「Start」に変更しておきます。

```
Command1.Caption = "Start"
```

「Else」以降には、「Start」ボタンが押されたときの動作を記述しておきます。上記の「Stop」ボタンの動作とまったく逆の処理を行います。つまり、スロットの動作を開始させるために［Interval］に「100」を入力し、変数Modeを「True」に設定し、コマンドボタンの［Caption］プロパティを「Stop」に変更するわけです。

```
Else
    Timer1.Interval = 100
    Mode = True
    Command1.Caption = "Stop"
end If
```

5 UserControlの起動時の初期化処理は、UserControlオブジェクトのInitializeイベントプロシージャで行います。これは、フォームのLoadイベントプロシージャと同じような使い方をします。

コントロールでは、2つの変数を初期化するために使用しています。変数Flagは、コントロールデザイン時に、Image1(0)が最初に表示されるように、［Visible］プロパティを設定しました。したがって、2番目のイメージImage1(1)から動作するように設定しています。

```
Private Sub UserControl_Initialize()
    Flag = 1
    Mode = True
End Sub
```

6 以上で、コントロールが完成です。コントロールの配置対象がフォームからUserControlというオブジェクトに変わっただけで、あとの操作は標準EXEプログラムの作成とまったくいっしょです。

次のレッスンでは、新しいフォームにこのコントロールを組み込んで動作させる方法を紹介します。

まとめ

ActiveXコントロールを作成するのは、標準EXEファイルの作成とほとんど変わりません。単独で動作するプログラムとほとんど同じ機能をコントロール化して持っておくこともできます。

Lesson 37 (2)

第10日 2時限目

ActiveX コントロールを作る

Lesson37.vbp
VB6CD
Lesson37

× Learning Edition

ActiveX コントロールの組み込みとテスト

ActiveX コントロールを組み込む方法と動作を確認する方法を覚えます。

今回作成するプログラム

I:¥Program Files¥VB98¥Slot.html - Microsoft Internet Explorer - [オフライン...
Stop
トランプのマークがスロットマシンのように回転している
[Stop] ボタンをクリックするとマークの回転が止まり、ボタンが [Start] に変わる

I:¥Program Files¥VB98¥Slot.html - Microsoft Internet Explorer - [オフライン...
Start
[Start] ボタンをクリックすると、回転を再開する

POINT

前回のレッスンで作成したActiveX コントロール「SlotGame」を、Visual Basic 上で動作させます。
また、コントロールの生成を行います。

動作テストの操作

※「SlotGame.vbp」をVisual Basicで読み込んでください。

1 ［標準］ツールバーの［開始］ボタンをクリックする

1 ここをクリック

2 最初だけ［プロジェクトプロパティ］ダイアログが表示されるので、［OK］ボタンをクリックする

1 ここをクリック

3 ブラウザが起動し、コントロールが動作する

ブラウザでActiveXコントロールが動作する

「SlotGame.vbp」

このファイルは、付録CD-ROMの「Lesson37」フォルダに収録されています。

ダイアログは最初だけ

このダイアログボックスは、次回からは表示されません。

動作の確認

プログラムを実行し、フォーム上の［Stop］ボタンをクリックすると、その上で回転しているトランプのマークが止まり、ボタンが［Start］に切り替わります。［Start］ボタンをクリックすると、マークの回転が再開されます。

4 Visual Basicに戻り［標準］ツールバーの［終了］ボタンをクリックして終了する

1 ここをクリック

組み込みの操作

1 ［ファイル］メニューから［SlotGame.ocxの作成］を選ぶ

1 ここをクリック

2 ここをクリック

2 ［実行可能ファイルの作成］ダイアログで、［オプション］をクリックし、バージョン情報を設定する

1 ［オプション］をクリック

SlotGame - プロジェクト プロパティ

1 バージョン情報を入力

2 [OK] をクリック

3 新しいプロジェクトを追加する

Project1 - Form1 (Form)

4 ツールボックスにSlotGameコントロールを組み込む

コンポーネント

1 ここをチェック

2 ここをクリック

バージョン情報の設定

設定方法の詳細については、Lesson36を参照してください。

新しいプロジェクトの追加

[標準] ツールバーの [プロジェクトの追加] ボタンをクリックし、[標準EXE] をクリックします。

標準EXEを追加すると [プロジェクトグループ] ウィンドウの様子は以下のようになっています。

プロジェクト グループ - Group1

[コンポーネント] ダイアログ

[コンポーネント] ダイアログボックスを表示するには、ツールボックスをマウスの右ボタンでクリックし、ショートカットメニューで [コンポーネント] をクリックします。

ツールボックスの変化

ActiveXコントロールのデザインウィンドウが開いているときは、ツールボックスのコントロールボタンは使用できないようになっています。

デザインウィンドウを閉じると使用可能になります。

ツールボックスにSlotGameが組み込まれる（ボタン名は［Slot］）

5 ツールボックスで［Slot］ボタンをクリックし、フォーム上にSlotコントロールを配置する

1 ここをクリックして…

2 そのままここまでドラッグ

6 ［標準］ツールバーの［開始］ボタンをクリックして、プログラムを実行させる

解説

1 Visual Basic 5.0では、作成したActiveXコントロールの、単独での動作チェックはできませんでした。そのため、コントロールの動作確認をするために、プロジェクトに標準EXEプロジェクトを追加し、このフォームにコントロールを組み込んで動作させていました。しかし、Visual Basic 6.0からは、作成したActiveXコントロールの動作テストは、ブラウザを使ってHTMLに組み込んだ状態での動作チェックができるようになりました。

コントロールを作成して、［標準］ツールバーの［開始］ボタンをクリックすると、一番はじめだけ「デバッグ」の設定ウィンドウが開きます。ここで、ブラウザでの表示設定を確認して［OK］ボタンをクリックすれば、そのままインターネットエクスプローラでActiveXコントロールの動作チェックが行えます。

2 作成したActiveXコントロールを他のプログラムに組み込むには、一度コントロールの作成を実行します。

ActiveXコントロールを、外部ファイル（.OCX）として生成するには、EXEプログラムの作成と同様、Visual Basicの［ファイル］メニューから選択して実行します。このとき、メニューの名前が「xxx.ocxの作成」に変わっています。メニューを選んだら、あとはEXEファイルの作成と同様に、バージョン番号やバージョン情報を組み込んで、作成を実行します。

ActiveXコントロールを作成したら、新しいプロジェクトを標準EXEで用意します。そして、ツールボックスの「コンポーネント」で、そのコントロールを追加し、フォームに配置します。

まとめ

ActiveX コントロールのファイル名は、初期設定ではプロジェクトファイルの名前が使われます。作成したocx ファイルは、Visual Basic のツールボックスに組み込むことができるので、他のプログラムからも自由に使うことができるようになります。

ワンポイントアドバイス

作成したActiveX コントロールをフォームに配置するときは、必ずUserControl の編集（デザイン）ウィンドウを閉じてから行ってください。作成中のActiveX コントロールが表示されていると、フォームのツールパレットには、そのコントロールは表示されません。

Lesson 38

第10日
3時限目

ActiveX
コントロール
を作る

Lesson38.vbp
VB6CD
Lesson38

×Learning Edition

ActiveX コントロールを HTML に組み込む

ActiveX コントロールの HTML への組み込み方法を覚えます。また、ActiveX コントロールを HTML ファイルとともに Web で公開する方法を知ります。

今回作成する HTML ファイル

New Page - Microsoft Internet Explorer
ファイル(F) 編集(E) 表示(V) 移動(G) お気に入り(A) ヘルプ(H)
アドレス http://www.big.or.jp/~seto/SlotGame.htm
リンク Internet Explorer ニュース
スロットゲーム
Stop
Stopボタンを押すと、停止します。
ハート:超ラッキー
スペード:ややラッキー
ダイヤ:アンラッキー
クラブ:超アンラッキー
インターネット ゾーン

POINT

作成した ActiveX コントロール「SlotGame.ocx」を、インターネットエクスプローラ 4.01 に付属する FrontPage Express を使って、HTML 文書に組み込みます。

HTML 文書では、他の HTML タグといっしょに記述できます。

あらかじめ、ActiveX コントロールをコンパイルしておいてください。

操作方法

1 FrontPage Express を起動する

2 次のような HTML を作成する

FrontPage Express

FrontPage Expressは、Microsoft社が無償で提供している、インターネットエクスプローラ4.01に同梱されている、HTML作成ソフトです。このソフトを使えば、簡単にActiveXコントロールをHTMLに組み込むことができます。

FrontPage Expressを起動するには、タスクバーの［スタート］メニュー→［プログラム］→［Internet Explorer］→［FrontPage Express］を順にクリックします。

HTML の作成

HTMLを作成する方法については、後述の「解説」を参照してください。

3 ［挿入］メニュー→［ほかのコンポーネント］→［ActiveX コントロール］を順にクリックする

1 ここをクリック

4 リストボックスで［SlotGame.Slot］を選択し、［OK］をクリックする

1 ここをクリック

2 ［OK］ボタンをクリック

HTML にコントロールが挿入される

「SlotGame.Slot」

あらかじめActiveXコントロールをコンパイルしていないと、「SlotGame.Slot」はこのリストには表示されません。

スクロールバー

目的のコントロールが見つからないときは、スクロールバーをスクロールして探してください。

5 HTML を「Slot.htm」という名前で保存する

1 保存先を指定

2 名前を入力

3 ［保存］をクリック

6 保存した HTML を WWW ブラウザで開く

1 ファイルのパスを指定

HTML の保存

ツールバーの［上書き保存］ボタンか、［ファイル］メニューの［上書き保存］をクリックします。

WWW ブラウザ

HTML ファイルや、インターネットでホームページを見るソフトを WWW ブラウザと呼びます。

ActiveX コントロールを表示できるブラウザは、インターネットエクスプローラ（バージョン 3.0 以降）だけです。Netscape Navigator では表示できませんので、注意してください。

解説

1 作成したActiveXコントロールを、HTMLに組み込むには、<object>というタグを使用します。このタグは、オプションに「id」と「classid」を持ちます。

これらは、<object>タグで組み込まれたものが、ActiveXコントロールであることをブラウザに認識させるために付けられるものです。そして、これらは私たちが勝手に付加するものではないため、ActiveXコントロールをHTMLに組み込むには、専用のツールが必要になります。

FrontPage Expressは、Microsoft社が自社のホームページで無償で提供している、ActiveXコントロールをHTMLに組み込むための専用のエディタです。このソフトを使えば、簡単にActiveXコントロールをHTMLに組み込むことができます。

```
<object classid="clsid:5E9BED5A-4B8F-11D2-8C3A-525400DAEBA0"
    align="baseline" border="0" width="320" height="240">
</object>
```

2 <object>タグは、通常のHTMLタグです。したがって、HTMLに組み込まれたActiveXコントロールも、JPEGイメージなどと同じようにHTMLタグを使ってレイアウトすることができます。また、ActiveXコントロールを表示するページは、HTMLタグを使って装飾することができます。

```
<html><head>
<meta http-equiv="Content-Type"content="text/html; charset
    =x-sjis">
<meta name="GENERATOR" content="Microsoft FrontPage Express
    2.0">
<title>New Page</title>
</head>

<body bgcolor="#C0C0C0">
<div align="center"><center>
<table border="2" cellpadding="8">
    <tr>
        <td bgcolor="#808000">
        <font color="#FFFF00" size="6">スロットゲーム</font>
        </td>
    </tr>
</table>
</center></div>
<p align="center">
<object classid="clsid:5E9BED5A-4B8F-11D2-8C3A-525400DAEBA0"
    align="baseline" border="0" width="320" height="240">
</object></p>
```

この値は状況により変化します

```
<p align="center">Stopボタンを押すと、停止します。</p>
<div align="center"><center>
```

3

一度HTMLにActiveXコントロールを組み込んだ後に、Visual Basicを使ってActiveXコントロールをコンパイルし直した場合は、ActiveXコントロールパッドでHTMLへの組み込みをやり直す必要があります。これは、コントロールをコンパイルするたびに「classid」の値が変化するからです。この、再組み込み作業を行わないと、ブラウザはActiveXコントロールを正常に表示できなくなります。

作成したActiveXコントロールと、HTMLに組み込んだActiveXコントロールは、完全に同じ物になるようにしてください。

また、Visual BasicでActiveXコントロールを変更した場合、WWWブラウザなどでそのActiveXコントロールを開いていると、同じファイル名でActiveXコントロールが生成できなくなります。必ず、同じ名前のActiveXコントロールが使用されていないことを確認してから、ActiveXコントロールの生成作業を行ってください。

4

独自に作成したActiveXコントロールを、組み込んだHTMLとともにWebで公開する場合は、コントロールの所在を明記しておく必要があります。

ActiveXコントロールを組み込んだHTMLがブラウザで読み込まれると、ブラウザは使われているActiveXコントロールを自動的にダウンロードします。このとき、コントロールの所在が明記されていないと、コントロールのダウンロードに失敗し、ActiveXコントロールがブラウザに表示されなくなってしまいます。

コントロールの所在は、<object>タグの［CodeBase］プロパティを使用します。これは、FrontPage Expressのプロパティウィンドウで設定できます。

①FrontPage Expressを起動して、ActiveXコントロールを組み込んだHTMLを表示します。

②ActiveXコントロールの上でマウスの右ボタンを押し、ポップアップメニューから［ActiveXコントロールのプロパティ］を選びます。

1 ここを右クリック
2 ここをクリック

③プロパティウィンドウにある、「コードソース」の欄に、HTMLを公開するWebサーバのURLを、httpから記述します。このとき、ActiveXコントロールのファイル名（SlotGame.ocx）まできちんと記述してください。パス名だけでは正常にダウンロードされません。また、ファイル名の大文字小文字も正確に記述し、サーバ側のファイル名ときちんと一致するようにしてください。以下は著者のホームページのサンプルアドレスです。

```
http://www.big.or.jp/~seto/SlotGame.ocx
```

これは著者のホームページのアドレスなので、必要に応じて自分のアドレスに変更する

④プロパティの編集画面を閉じ、FrontPage Expressで［表示］メニューの［HTML］を選びます。

1 ここをクリック
2 ここをクリック

⑤HTMLの<object>タグに、新たに「CODEBASE」が追加されています。

```
CODBASE = "http://www.big.or.jp/~seto/SlotGame.ocx"
```

ここが追加されている

確認したらここを押す

⑥作成したHTMLファイルとActiveXコントロール（ここでは、「SlotGame.ocx」）を、WebサーバにFTPソフトで転送します。このとき、「CODEBASE」で指定したURLと必ず一致するディレクトリに転送してください。

FTPソフト

FTPソフトで転送する

⑦ブラウザでHTMLファイルを読み込み、動作テストをします。このとき、インターネットエクスプローラの［インターネットオプション］メニューで、セキュリティのページにある［セキュリティレベル］を「低」に設定してください。
独自に作成したActiveXコントロールには、「デジタル署名」がありませんので、セキュリティレベルが「中」以上だと、ブラウザがダウンロードしません。

インターネット オプション

全般 セキュリティ コンテンツ 接続 プログラム 詳細設定

Web コンテンツのゾーンごとに別のセキュリティ レベルを設定できます。

ゾーン(Z): インターネット ゾーン リセット(R)

インターネット ゾーン

このゾーンには、ほかのゾーンに設定していないすべての Web サイトが含まれています。 サイトの追加(D)...

このゾーンのセキュリティ レベル:

高 (最も安全)(H) コンピュータに損害を与える可能性があるコンテンツは除外されます。

中 (安全)(M) 損害を与える可能性があるコンテンツの実行前に警告します。

低(L) 損害を与える可能性があるコンテンツの実行前に警告しません。

カスタム (詳しい知識のあるユーザー向け)(C) セキュリティは選択する設定に基づきます。 設定(S)...

OK キャンセル 適用(A)

1 [低] をクリック

2 [OK] をクリック

⑧ダウンロードを開始すると、途中でインストールの確認が表示されるので、「はい」をクリックするとダウンロードを完了し、ActiveXコントロールが動作を開始します。

ダウンロードを開始

スロットゲーム

Stopボタンを押すと、停止します。

セキュリティの警告

"http://www.big.or.jp/~seto/SlotGame.ocx" をインストールして実行しますか?

発行者を決定できません。原因:

AUTHENTICODE 署名が見つかりません。

はい(Y) いいえ(N) 詳細情報(M)

途中でインストールの確認が表示される

ダウンロードを完了し
動作を始める

まとめ

ActiveX コントロールを HTML に組み込む場合は、FrontPage Express や ActiveX コントロールパッドなどの、専用のエディタを使って組み込みます。また、コントロールの作成バージョンが変わったときは、HTML への組み込みもやり直す必要があります。

HTML に ActiveX コントロールを組み込むと、これまでのホームページにはない、ユーザー対話型でダイナミックな Web コンテンツを作成できるので、自分のホームページをとても魅力的なページに変えられます。

しかし、一方では ActiveX コントロールを表示できるブラウザが、1998 年 11 月現在インターネットエクスプローラしかない状況もありますので、ページに「インターネットエクスプローラで見てください」などの断り書きを添えるようにするといいでしょう。

また、レッスンを終えたら、インターネットエクスプローラの［セキュリティ］設定を元に戻しておきましょう。

Let's VBSchool!!

付録
各種リファレンス

Visual Basicプログラミングに役立つ各種リファレンスを掲載しました。

わからない言葉や関数があれば用語集や関数リファレンスを参照してください。ASCII文字コード一覧はChr関数などを使用するときに利用できるものです。データ型一覧はVisual Basicで扱えるデータ型を知っておくためのものです。エラー番号一覧はデバッグ作業に有効です。Visual Basicで生成されるファイルの知識があればファイルを配布するときに役立つでしょう。ショートカット一覧でショートカットを使いこなせば作業を効率化することができます。

巻末に各レッスンの練習問題の解答が付いています。

●付録

❏ 用語集
❏ 関数リファレンス
❏ ASCII文字コード一覧
❏ データ型一覧
❏ エラー番号一覧（一部）
❏ プロジェクトファイルの形式
❏ ショートカット一覧
❏ 練習問題の解答

用語集

レッスン中に使用した用語をまとめました。これ以外の用語については Visual Basic のオンラインヘルプ「MSDN ライブラリ」も参照してください。

※この用語集は以下のような構成になっています。

（例） MouseMove イベント・・・・・・・・・・・・・・・・用語
　　マウスを動かすと発生するイベント。・・・・・・用語の説明

英字

API
Application Programming Interface の略で、Windows の持っている機能をプログラムで利用するために用意されている機能。Windows プログラムのほとんどが、何らかの形でこの API を呼び出しながら動作している。

ActiveX コントロール
Microsoft が開発した新しい技術「ActiveX」テクノロジーを組み込んだコントロールのこと。これまで、Visual Basic のコントロールは Visual Basic だけでしか使用できなかったが、ActiveX コントロールになってからは、Visual C++ や Visual J++、ExcelVBA など、複数のアプリケーションや開発環境から使うことができる。

DAO
Data Access Object の略で、データベースを操作する機能を組み込んだオブジェクト。Visual Basic や VBA で使うと、簡単にデータベースの操作ができる。

DLL ファイル
Dynamic Link Library の略で、プログラムから呼び出されるファイル。プログラムの機能をプログラムに埋め込まず、この DLL ファイルに収録し、プログラムの本体から呼び出して使うことができる。DLL ファイル自身は、単独で実行できない。

DHTML
Dynamic Hyper Text Markup Language の略。ホームページ作成言語である HTML を更に機能拡張し、動きのあるページや、画像と文字の重ねあわせなど、デザイン機能が強化されている。また、スクリプト言語によるプログラム機能も組み込むことができる。

HTML
Hyper Text Markup Language の略。ホームページを作成するために作られた文書作成言語。ページ間を簡単に移動できる「ハイパーテキスト」機能や、文字の拡大・装飾、作表、画像組み込みなど、マルチメディア的な文書を作成できる。

MIDI ファイル
コンピュータ・ミュージック用にフォーマットされたファイル。録音・再生には「MIDI シーケンサ」と呼ばれる専用の機器や「MIDI 音源」が必要だが、Windows ではソフト的に再生できるようになっている。

MouseDown イベント
マウスのボタンを「押したとき」に発生するイベント。Click イベントと違い、マウスのボタンを押した時点で発生する。

MouseMove イベント
マウスを動かすと発生するイベント。

MouseUp イベント
マウスのボタンを「離したとき」に発生するイベント。

OLE
Object Linking & Embeddingの略で、日本語にすると「オブジェクトの連結と組み込み」という意味。Windowsの機能の1つで、オブジェクト同士を連動させる機能。

RGB
Red/Green/Brueの3つの色の組み合わせのこと。モニタの色表示に使用される。

WAVオーディオ
Windows用にフォーマットされたサウンドファイルの形式。

Webコンテンツ
WWWで扱われている情報の中身。HTML文書やダウンロードファイルなど、WWWで取り扱われるものを総称してこう呼ぶ。

WWW
World Wide Webの略。インターネットで利用できるサービスの1つで、ホームページの閲覧ができる。

ア行

イメージコントロール
イメージファイルを表示するコントロール。

インターネット
地球規模のネットワークのことで、電子メールやWWW、FTPなどのサービスを利用できる。

インタプリタ
プログラムのソースコードを読み込みながら、プログラムを実行する機能のこと。Visual Basicのコードをマシン語に翻訳しながら動作させるため、Visual Basicが動作していないと、プログラムを実行できない。

エラー処理
ユーザーの誤操作や、式の実行結果によって生じるプログラムの動作エラーを、あらかじめ予想して回避するための処理のこと。

演算子
計算に使用する記号で、式はデータと演算子で組み立てられる。Visual Basicで用意されている演算子には、四則演算を行う「算術演算子」や、データを比較する「比較演算子」などがある。

オブジェクト
フォームやコントロールなど、プログラムを構成するものすべてを、オブジェクトと呼ぶ。

オプションボタンコントロール
複数の項目から1つだけ選択してもらう場合に使用するコントロール。

カ行

構造化プログラミング
プログラムを機能ごとにまとめ、それぞれを呼び出しながらプログラムを組み立てていく手法。

コモンダイアログボックスコントロール
Windowsの機能の1つで、「共通ダイアログボックス」と呼ばれている。[ファイルを開く][名前を付けて保存]などのダイアログボックスは、これまで自分で作成しなければならなかったが、コモンダイアログボックスコントロールが導入されてからは、あらかじめ用意されているこれらのダイアログボックスを呼び出すだけで、簡単に使用できるようになった。

コントロール
コマンドボタンやリストボックスなど、プログラムを構成する部品のことをいう。動作する機能によって、さまざまなコントロールが用意されている。

コントロール配列
1つのコントロールをいくつも複製して使用すると、コントロールを配列として使用できる。1つのオブジェクト名に番号を付けて参照する。

コンパイラ
プログラムを、ソースコードから実行可能なファイルに変換する機能を「コンパイル」と呼び、この機能を持ったプログラムを「コンパイラ」という。

コンボボックスコントロール
テキストボックスコントロールとリストボックスコントロールを組み合わせたコントロール。リストからの選択とユーザーの入力のどちらも受け付けることができる。

サ行

シーケンシャル書き出し
ファイルの先頭から順番に書き込まれる方法のこと。

シェアウェア
有料で配布されるソフトウェアのこと。市販のソフトと違い、パッケージに入って店頭販売されるのではなく、インターネットやパソコン通信、雑誌の付録などを使って流通している。 無料で入手し、とりあえず使ってみて、気に入ったらお金を払う、というシステムになっている。

スクロールバーコントロール
ボタンを押すとバーが移動するコントロール。水平と垂直の２種類が用意されている。大きすぎる画面内容を表示するのに使用する。

ステップ実行
プログラムをソースコード１行ずつ実行する機能。

セキュリティ
「保安」「警備」などの意味で、コンピュータの世界では、コンピュータウィルスや外部からの不当な侵入を防ぐための機能のことをこう呼んでいる。

宣言セクション
モジュールの中で、変数を宣言するためだけに使用できるセクション。ここで宣言した変数は、モジュール内のすべてのプロシージャから参照できる。

ソースコード
プログラムの元になるリスト。

タ行

タイマーコントロール
ミリ秒単位でイベントを発生するコントロール。アニメーション効果などを設定する際に使用すると効果的。

タブ付きダイアログボックスコントロール
Windowsから導入された、小さなタブが付いたダイアログボックスを操作するコントロール。

チェックボックスコントロール
ユーザーがクリックすると、□内にチェックが付くコントロール。条件のオンオフを操作する場合に使用する。

著作権
ものを作成した権利。日本の法律では、プログラムといえども製作者が著作権を保有している。したがって、著作権者に無断で販売や複製を行うと、法律で罰せられる。

ツールチップ
メニューやツールバーのボタンの上に、マウスのポインタを持ってくると現れる、これらの内容を説明する小さな表示のこと。

ディレクトリリストボックス
指定したドライブのディレクトリ（フォルダ）名を一覧表示するコントロール。

データコントロール
Microsoft Accessなどの、データベースファイルを操作するためのコントロール。

テキストファイル
文字だけで構成されるファイル。

テキストボックスコントロール
文字の入力を受け付けるコントロール。プロパティの設定で複数行の編集やスクロールバーの有無を設定できる。

デジタル署名
ActiveX コントロールなどに組み込む、作成者名などが表示される電子署名。

デバイス
コンピュータに接続される機器のことをデバイスという。また、このデバイスとコンピュータを接続するために使用される、小さなシステム用ソフトを「デバイスドライバ」という。

デバッグ作業
プログラムの中に存在する不具合を「バグ」といい、このバグを見つけ出し修正する作業を「デバッグ」と呼ぶ。

トップレベルメニュー
最上位に位置するメニュー項目のこと。

ドライブリストボックス
コンピュータで利用可能なドライブをリスト表示するコントロール。

ドラッグ＆ドロップ
マウスのボタンを押しながらアイコンなどを移動させることを「ドラッグ」という。また、そのまま他のアイコンやオブジェクトに重ねる操作を「ドラッグ＆ドロップ」と呼んでいる。

ハ行

バージョン番号
作成したプログラムに、一連番号を付けることができる。これが、「バージョン番号」と呼ばれるもので、Visual Basic では自動的にこの番号を付加する機能を装備している。

比較演算子
２つのデータを比較して、答えを求める演算子。「=」は、２つのデータを比較し、等しければ真を返す。他に大なり（>）小なり（<）などがある。

ビットマップファイル
イメージファイルの１つで、Windows 専用のフォーマットで保存される。

標準モジュール
イベントプロシージャとは別に、独自のプロシージャを作成する場合に使用できる、コード専用のモジュールのこと。

ファイルリストボックス
指定したディレクトリ（フォルダ）にあるファイルを一覧表示するコントロール。

ファイル番号
開いたファイルを管理するために使用される番号。ファイル管理の操作では、ファイル名ではなくファイル番号で操作される。

フォーム
プログラムのベースとなるウィンドウのこと。プログラムによって、複数のフォームを持つことができる。

フォームテンプレート
フォーム作成に、あらかじめ用意された「バージョン番号表示」や「タイトル表示」など、フォームの雛形を使うことができるようになった。これを、フォームテンプレートという。

ブラウザ
ブラウズは「拾い読み」とか「ざっと見る」という意味。ホームページを見るソフトを「WWW ブラウザ」と呼ぶ。

フリーウェア
無償で提供されるソフトウェア。主に、インターネットやパソコン通信、雑誌の付録などで流通している。

ブレークポイント
プログラムのデバッグや動作テストに使用するVisual Basicの機能の1つで、ソースコードの任意の1行に目印を付けて、そこで動作をいったん停止させることができる。この目印を、ブレークポイントという。

フレームコントロール
コントロールをまとめたり、フォームに枠を作成するためのコントロール。

フロー制御
プログラムの流れを制御すること。特定の条件で以降の処理を振り分ける場合に使用される。Visual Basicでは、Ifステートメントや Select Caseステートメントが代表的なステートメント。

変数
プログラム実行時に、データを一時的に保管するために使われる「入れ物」のこと。中に入れるデータによって、変数の型が決められている。

変数のウォッチ
デバッグ機能の1つで、プログラムを実行しながら変数の中身を調べることができる。

変数を宣言する
変数を使用する場合は、あらかじめこういう変数を使う、という宣言をすることができる。その場合は、Dimステートメントで変数名を記述し、データ型を指定して宣言する。

ポップアップメニュー
フォームの上でマウスのボタンを押すと表示されるメニュー。Windowsと、Windows対応のアプリケーションでは、ほとんどこのメニュー機能を組み込んでいる。

マ行

マウスイベント
マウスの操作を行った時に発生するイベントで、マウスを動かしたときやボタンをクリックしたときなど、数種類のイベントが用意されている。

マルチメディアコントロール
サウンドファイルの再生や、AVIビデオファイルなど、マルチメディア関連のデータファイルを再生するためのコントロール。

メソッド
コントロールに対し、なにか動作を行わせる場合に使用される、Visual Basicの機能の1つ。

メッセージボックス
ユーザーから「はい」「いいえ」などの応答が必要なメッセージ、あるいはエラー、警告、注意などの短いメッセージを表示するためのダイアログボックス。

メディア
「媒体」「伝達手段」の意味。

メニューエディタ
プログラムにメニューを組み込むためのツール。

メモリ
コンピュータの中で、データを記憶しておく場所のこと。Visual Basicでは、直接メモリエリアを参照することができない。

モーダルとモードレス
フォームの表示形態で、そのフォームを閉じないと元のプログラムに戻れないのが「モーダル表示」、どちらも操作できるのが「モードレス表示」と呼ぶ。

モジュール
1つのかたまりのことで、プログラムの世界では、プログラムの機能を部分的にまとめたものをモジュールと呼んでいる。

文字列
複数の文字を組み合わせたものを、「文字列」と呼んでいる。

ヤ行

ユーザーインターフェイス
プログラムの、ユーザーと接する部分を表す言葉。通常はボタンやメニューのことを指す。

ラ行

ランタイムDLL
DLLファイルの一種で、プログラムを実行する上で必要なDLLファイルのこと。

ループ処理
一定の条件や回数で、1つの処理を何回も繰り返し実行する処理。Visual Basicでは、For...Nextステートメントや、Do...While...Loopステートメントなどが使われる。

レジストリ
Windowsでは、アプリケーションの登録や設定を「システムレジストリ」というファイルに記録する。Visual Basicでは、自分で作成したプログラムで、このレジストリを使うことができる。

連結コントロール
データベースコントロールといっしょに連動して使用できるコントロールのことで、プロパティの設定だけで連動できる。

論理演算子
OR(または)、AND(及び)、NOT(でない)という論理演算に使用する演算子。

論理型(Boolean)データ
Visual Basicで扱われるデータ型の1つで、真(True)と偽(False)の2つのデータしか持たない、論理型のデータ。

ワ行

ワイルドカード
トランプでいうと「万能札」と同じ役割を果たす機能。ファイル名の指定時に、すべての種類のファイルを対象とする場合は、「*.*」と記述する。

関数リファレンス

Visual Basicで使用できる関数の一覧です。これらの関数の詳細については、Visual Basicのオンラインヘルプ「MSDNライブラリ」も参照してください。

※この関数リファレンスは以下のような構成になっています。

（例） **Abs**・・・・・・・・・・・・・・・・・・・・・・・・・・・関数名
数値の絶対値を同じデータ型で返す ・・・・関数の説明
Abs(数値) ・・・・・・・・・・・・・・・・・・・・・・関数の書式

A

Abs
数値の絶対値を同じデータ型で返す。

Abs(数値)

Array
配列が格納されたバリアント型の値を返す。

Array(値1, 値2, 値3,...)

Asc
文字列内の先頭の文字を文字コード（整数）に変換する。

Asc(文字列)

AscB
文字列に含まれた最初のバイトデータを取得する。

AscB(文字列)

AscW
Unicode文字セットの文字を返す。

AscW(文字列)

Atn
指定した数値のアークタンジェントを倍精度浮動小数点数型 (Double) で返す。

Atn(倍精度浮動小数点数型 (Double) の数値または任意の数式)

C

CBool
式または数値をブール型に変換する。

CBool(式または数値)

CByte
式または数値をバイト型に変換する。

CByte(式または数値)

CCur
式または数値を通貨型に変換する。

CCur(式または数値)

CData
式または数値を日付型に変換する。

CData(式または数値)

CDbl
式または数値を倍精度浮動小数点数型に変換する。

CDbl(式または数値)

CDec
式または数値を10進型に変換する。

CDec(式または数値)

Choose
引数のリストから値を選択して返す。

Choose(引数 index, 選択可能な値のリスト)

Chr
指定した文字コードに対応する文字を示す文字列を返す。

Chr(文字コード)

CInt
式または数値を整数型に変換する。

CInt(式または数値)

CLng
式または数値を長整数型に変換する。

CLng(式または数値)

Command
実行可能なプログラムを起動させるために使用するコマンラインの引数の部分を返す。

Command

Cos
指定した角度のコサインを倍精度浮動小数点数型 (Double) で返す。

Cos（角度をラジアン単位で指定)

CreateObject
ActiveX オブジェクトへの参照を作成して返す。

CreateObject(クラス名)

CSng
式または数値を単精度浮動小数点数型に変換する。

CSng(式または数値)

CStr
式または数値を文字列型に変換する。

CStr(式または数値)

CurDir
指定したドライブの現在のパスを返す。

CurDir(ドライブ名)

CVar
式または数値をバリアント型に変換する。

CVar(式または数値)

CVErr
ユーザーが指定したエラー番号を、エラー値に変換した値を返す。

CVErr(エラー番号)

D

Date
現在のシステムの日付を取得する。

Date

DateAdd
指定された時間間隔を加算した日付を返す。

DateAdd(追加する時間間隔の形式, 追加する時間間隔の数, 時間間隔を追加する日付)

DateDiff
２つの指定した日付の時間間隔を返す。

DateDiff(間隔を計算するための時間単位, 間隔を計算する日付, 間隔を計算する日付[, 週の始まりの曜日を表す定数 [, 年度の第１週を表す定数]])

DatePart
日付を評価し、特定の時間間隔部分を取得する。

DatePart(時間間隔の単位, 評価する値[,週の始まりの曜日を表す定数[, 年度の第１週を表す定数]])

DateSerial
指定された年、月、日に対応する日付を取得する。

DateSerial(年, 月, 日)

DateValue
文字列を日付に変換する。

DateValue(日付)

Day
指定された日付の“日”で表される部分を取得する。

Day（日付）

DDB
倍率法などの指定した方法を使って特定の期における資産の減価償却費を返す。

DDB(資産を購入した時点での価格，耐用年数が終了した時点での資産の価格，資産の耐用年数，減価償却費を求める期を示す値[，減価償却率])

Dir
指定したパターンやファイル属性と一致するファイルまたはフォルダの名前を返す。

Dir[(ファイル名[，取得するファイルが持つ属性の値の合計])]

DoEvents
プログラムで占有していた制御をオペレーティングシステムに渡すフロー制御関数。

DoEvents( )

E

Environ
オペレーティング システムの環境変数を返す。

Environ({環境変数 | 環境文字列の順番})

EOF
ファイルの終端に達したかどうかを返す。

EOF(ファイル番号)

Error
指定したエラー番号に対応するエラーメッセージを返す。

Error[(エラー番号)]

Exp
指数関数 (e を底とする数式のべき乗) を計算する数値演算関数 Exp(数値)。

Exp(式または数値)

F

FileAttr
Open ステートメントで開いたファイルのファイルモードを返す。

FileAttr(ファイル番号，取得するファイル情報の種類)

FileDateTime
指定したファイルの作成日時または最後に修正した日時を返す。

FileDateTime(ファイル名)

FileLen
ファイルのサイズをバイト単位で返す。

FileLen(ファイル名)

Filter
指定されたフィルタ条件に基づいた文字列配列のサブセットを含むゼロベースの配列を返す。

Filter(検索先文字列，検索する文字列[，含まれるかどうか[，文字列比較モード]])

Fix
指定した数値の小数部分を取り除いた整数値を返す。

Fix(数値)

Format
式を指定した書式に変換する。

Format(式[，定義済み書式または表示書式指定文字[，週の1日目を指定する定数[，年の第1週を指定する定数]]])

FormatCurrency
システムの [コントロール パネル] で定義されている書式を使って通貨形式の文字列を返す。

FormatCurrency(変換する式[,小数点以下の桁数 [,小数点の左側のゼロを表示するかどうか [,負の値をかっこで囲むかどうか [, [地域のプロパティ] で指定されている桁区切り記号を使用するかどうかを表す定数]]]])

FormatDateTime
日付形式または時刻形式の文字列を返す文字列処理関数

FormatDateTime(書式を変換する日付式[,使用されている日付/時刻形式を表す数値])

FormatNumber
数値形式の文字列を返す。

FormatNumber(変換する式[,小数点以下に表示する桁数 [,小数点の左側のゼロを表示するかどうか [,負の値をかっこで囲むかどうか [, [地域のプロパティ] で指定されている桁区切り記号を使用するかどうかを表す定数]]]])

FormatPercent
100 で乗算したパーセント形式の式にパーセント記号 (%) を付加して返す。

FormatPercent(変換する式[,小数点以下に表示する桁数 [,小数点の左側のゼロを表示するかどうか [,負の値をかっこで囲むかどうか [, [地域のプロパティ] で指定されている桁区切り記号を使用するかどうかを表す定数]]]])

FreeFile
使用可能なファイル番号を整数型 (Integer) の値で返す。

FreeFile[(ファイル番号の範囲)]

FV
定額の支払いを定期的に行い、利率が一定であると仮定して、投資の将来価値を返す。

FV(利率, 支払い回数の合計, 毎回の支払額[, 支払いの合計金額[, 支払期日]])

G

GetAllSettings
Windows のレジストリにあるアプリケーションの項目から、すべてのキー設定および各キー設定に対応する値のリストを返す。

GetAllSettings(キー設定を取得するアプリケーション名, キー設定を取得するセクション名)

GetAttr
ファイルまたはフォルダの属性を返す。

GetAttr(ファイル名)

GetAutoServerSettings
ActiveX コンポーネントの登録情報を返す。

object.GetAutoServerSettings([コンポーネントのプログラム ID], [コンポーネントのクラス ID])

GetObject
ファイルから取得した ActiveX オブジェクトへの参照を返す。

GetObject([ファイル名] [, オブジェクトのクラス])

GetSetting
Windows のレジストリにあるアプリケーションの項目から、キー設定値を返す。

GetSetting(キー設定を取得するアプリケーション名, キー設定があるセクション名, キー設定名[, 初期値])

H

Hex
指定した値を 16 進数で返す。

Hex(数値)

Hour
1日の時刻を表す 0～23 の範囲の整数を返す。

Hour(時刻)

I

IIf
式の評価結果によって、2つの引数のうち1つを返す。

IIf(評価対象の式, 式が真(True)の場合に返す値, 式が偽(False)の場合に返す値)

IMEStatus
IMEの現在の状態を返す。

IMEStatus

Input
開いたファイルから指定した文字数の文字列を読み込む関数。

Input(文字数, [#]ファイル番号)

InputB
テキストファイルに含まれているバイトデータを読み込む関数。

Input(バイト数, [#]ファイル番号)

InputBox
ダイアログボックスにメッセージとテキストボックスを表示し、テキストボックスの内容を返す。

InputBox(表示する文字列[, ダイアログボックスのタイトル] [, テキストボックスに既定値として表示する文字列] [, 水平方向の表示位置] [, 垂直方向の表示位置] [, 使用するヘルプファイルの名前, ヘルプトピックに指定したコンテキスト番号])

InStr
ある文字列の中から指定した文字列を検索し、最初に見つかった文字位置(先頭からその位置までの文字数)を返す。

InStr([検索の開始位置,]検索対象となる文字列式, 検索する文字列式[, 文字列比較の比較モードを指定する番号])

InStrRev
文字列の中から、指定された文字列を最後の文字位置から検索し、最初に見つかった文字の先頭からその位置までの文字数を返す。

InstrRev(検索先の文字列, 検索する文字列[, 各検索の開始位置[, 文字列比較のモード]])

Int
指定した数値の整数部分を取り除いた整数値を返す。

Int(数値)

IPmt
定額の支払いを定期的に行い、利率が一定であると仮定して、投資期間内の指定した期に支払う金利を返す。

IPmt(利率, 金利支払額を求める期, 支払い回数の合計, 支払いの合計金額[, 最後の支払いを行った後に残る現金の収支[, 支払期日]])

IRR
一連の定期的なキャッシュフロー(支払いと収益)に対する内部利益率を返す。

IRR(一連のキャッシュフローの値[, IRR関数の計算結果に近いと思われる値])

IsArray
変数が配列であるかどうかを調べ、結果をブール型(Boolean)で返す。

IsArray(配列変数名)

IsDate
式を日付に変換できるかどうかを調べ、結果をブール型（Boolean）で返す。

IsDate(式)

IsEmpty
変数がEmpty値かどうかを調べ、結果をブール型（Boolean）で返す。

IsEmpty(変数)

IsError
式がエラー値かどうかを調べ、結果をブール型（Boolean）で返す。

IsError(式)

IsMissing
プロシージャに省略可能なバリアント型（Variant）の引数が渡されたかどうかを調べ、結果をブール型（Boolean）で返す。

IsMissing(引数)

IsNull
式にNull値が含まれているかどうかを調べ、結果をブール型（Boolean）で返す。

IsNull(式)

IsNumeric
式が数値として評価できるかどうかを調べ、結果をブール型（Boolean）で返す。

IsNumeric(式)

IsObject
識別子がオブジェクト変数を表しているかどうかを示すブール型（Boolean）の値を返す。

IsObject(変数)

J

Join
配列に含まれる各要素の内部文字列を結合して作成される文字列を返す。

Join(結合する文字列を含む1次元配列[，文字列を区切るのに使用する文字])

L

LBound
配列の指定された次元で使用できる最小の添字を、長整数型の値で返す。

LBound(配列変数[，添字の最小値を調べる対象となる配列の次元])

LCase
アルファベットの大文字を小文字に変換する。

LCase(文字列)

Left
文字列の左端から指定した文字数分の文字列を返す。

Left(文字列，取り出す文字列の文字数)

LeftB
文字列の左端から指定したバイト数分の文字列を取り出す。

Left(文字列，取り出す文字列のバイト数)

Len
文字列の文字数または変数に必要なバイト数を返す。

Len(文字列または変数名)

LoadPicture
指定したグラフィックをロードする。

LoadPicture(ファイル名)

Loc
開いたファイル内の現在の読み込み位置または書き込み位置を返す。

Loc(ファイル番号)

LOF
Open ステートメントを使用して開いたファイルの長さをバイト単位で返す。

LOF(ファイル番号)

Log
倍精度浮動小数点数型の自然対数を返す数値演算関数。

Log(数値)

LTrim
指定した文字列から先頭のスペースを削除した文字列を返す。

LTrim(文字列)

M

Mid
文字列から指定した文字数分を返す。

Mid(文字列, 先頭からの文字数[,取り出す文字数])

MidB
文字列から指定した文字数分の文字列をバイト単位で返す。

Mid(文字列, 先頭からのバイト数[,取り出すバイト数])

Minute
時刻の分を表す 0 ～ 59 の範囲の整数を返す。

Minute(時刻)

MIRR
一連の定期的なキャッシュフロー (支払いと収益)に基づいて、修正内部利益率を返す。

MIRR(キャッシュフローの値, 支払額に対する利率, 収益額に対する利率)

Month
1 年の何月かを表す 0 ～ 12 の範囲の整数を返す。

Month(日付)

MonthName
指定された月を表す文字列を返す。

MonthName(月を示す数値[, 月名を省略するかどうかを表すブール値])

MsgBox
ダイアログ ボックスにメッセージを表示し、どのボタンがクリックされたのかを示す値を返す。

MsgBox(メッセージ[, ボタンの種類][, ウィンドウタイトル] [, ヘルプファイル名, ヘルプコンテキスト番号])

N

Now
コンピュータのシステムの日付と時刻の設定に基づいて、現在の日付と時刻を返す。

Now

NPer
定額の支払いを定期的に行い、利率が一定であると仮定して、投資に必要な期間を返す。

NPer(利率, 毎回の支払額, 現在の投資額または収益[, 現金の収支[, 支払期日]])

NPV
一連の定期的なキャッシュ フロー (支払いと収益) と割引率に基づいて、投資の正味現在価値を返す。

NPV(割引率, キャッシュ フローの値)

O

Oct
引数に指定した値を 8 進数で表すバリアント型 (内部処理形式 String の Variant) の値を返す。

Oct(数値)

P

Partition
ある数値が、区切られた複数の範囲のうち、どの範囲に含まれるかを示すバリアント型の文字列を返す。

Partition(評価する整数, 対象となる範囲全体の先頭となる整数,対象となる範囲全体の末尾となる整数 , 対象となる範囲全体を等分する各範囲の長さを示す整数)

Pmt
定額の支払いを定期的に行い、利率が一定であると仮定して、投資に必要な定期支払額を返す。

Pmt(利率, 支払回数, 合計金額[, 収支[, 支払期日]])

PPmt
定額の支払いを定期的に行い、利率が一定であると仮定して指定した期に支払われる元金を返す。

PPmt(利率, 元金支払額を求める期, 支払い回数, 合計金額[, 現金の収支[, 支払期日]])

PV
定額の支払いを定期的に行い、利率が一定であると仮定して、投資の現在価値を返す。

PV(利率, 支払い回数, 支払額[, 現金の収支[, 支払期日]])

Q

QBColor
指定した色番号に対応するRGBコードを表す長整数型の値を返す。

QBColor(色番号)

R

Rate
投資期間を通じての利率を返す。

Rate(支払い回数, 毎回の支払額, 支払いや収益[, 現金の収支[, 支払い期日[, 利率の推定値]]])

Replace
指定された文字列の一部を、別の文字列で指定された回数分で置換する。

Replace(置換する文字列を含む文字列式 , 検索する文字列, 置換する文字列[, 検索開始位置[, 置換する文字列数[, 文字列比較のモード]]])

RGB
色のRGB値を表す長整数型の値を返す。

RGB(red値, green値, blue値)

Right
文字列の右端から指定した文字数分の文字列を返す。

Right(文字列, 取り出す文字列の文字数)

Rnd
単精度浮動小数点数型 (Single) の乱数を返す。

Rnd[数値(省略可能)]

Round
指定された小数点位置で丸めた数値を返す。

Round(丸めを行う数式 [,丸めを行う小数点以下の桁数を表す数値])

RTrim
指定した文字列から末尾のスペースを削除した文字列を返す。

RTrim(文字列)

S

Second
時間の秒を表す0～59の範囲の整数を返す。

Second(時刻)

Seek
Open ステートメントを使用して開いたファイルの現在の読み込み位置または書き込み位置を返す。

Seek(ファイル番号)

Sgn
引数に指定した値の符号をバリアント型の値で返す数値演算関数。

Sgn(数値)

Shell
指定したプログラムを実行する。

Shell(プログラム名[,起動時のウィンドウスタイル])

Sin
指定した角度のサインを倍精度浮動小数点数型の値で返す数値演算関数。

Sin(角度（単位ラジアン）)

SLN
定額法を用いて、資産の１期当たりの減価償却費を返す。

SLN(資産を購入した時点での価格, 耐用年数を経た後での資産の価格, 資産の耐用年数)

Space
指定した数のスペースからなる文字列を返す文字列処理関数。

Space(数値)

Spc
Print # ステートメントまたは Print メソッドと共に使用し、指定した数のスペースを挿入するファイル入出力関数。

Spc(スペースの数)

Split
各要素ごとに区切られた文字列から１次元配列を作成する。

Split(区切り文字を含んだ文字列式[, 文字列の区切り文字[, 返す配列の要素数[, 文字列比較のモード]]])

Sqr
数式の平方根を倍精度浮動小数点数型の値で返す数値演算関数。

Sqr(数値)

Str
数式の値を文字列で表した値（数字）で返す文字列処理関数。

Str(数値)

StrComp
文字列比較の結果を返す。

StrComp(文字列1, 文字列2[, 文字列比較のモード])

StrConv
文字列を指定した形式に変換する。

StrConv(文字列, 変換形式)

StrReverse
指定された文字列の文字の並びを逆にした文字列を返します。

StrReverse(文字の並びを逆にする文字列)

String
指定した文字コードの示す文字、または文字列の先頭文字を、指定した文字数だけ並べた文字列を返す文字列処理関数。

String(文字数, 文字列の先頭文字)

Switch
式のリストを評価し、リストの中で真となる最初の式に関連付けられたバリアント型の値または式を返す。

Switch(式-1, 評価-1[, 式-2, 評価-2 ... [, 式-n,評価-n]])

SYD
定額逓減法を使って、指定した期の減価償却費を返す。

SYD(資産を購入した時点での価格，耐用年数を経た後での資産の価格，資産の耐用年数，減価償却費を計算する期)

T

Tab
Print # ステートメントまたは Print メソッドと共に使用し、次の文字の出力位置を移動するファイル入出力関数。

Tab[(桁数)]

Tan
指定した角度のタンジェントを倍精度浮動小数点数型の値で返す数値演算関数。

Tan(角度（単位ラジアン）)

Time
現在のシステムの時刻を表すバリアント型の値を返す。

Time

Timer
午前0時から経過した秒数を表す単精度浮動小数点数型の値を返す。

Timer

TimeSerial
引数で指定した時、分、および秒に対応する時刻を含むバリアント型の値を返す。

TimeSerial(時，分，秒)

TimeValue
時刻を表すバリアント型の値を返す。

TimeValue(時刻)

Trim
指定した文字列から先頭と末尾の両方のスペースを削除した文字列を返す。

Trim(文字列)

TypeName
変数に関する情報を提供する文字列型の文字列を返す。

TypeName(変数名)

U

UBound
配列の指定された次元で使用できる添字の最大値を、長整数型の値で返す。

UBound(配列名[，配列の次元])

UCase
指定したアルファベットの小文字を大文字に変換する文字列処理関数。

UCase(文字列)

V

Val
指定した文字列に含まれる数値を適切なデータ型に変換する。

Val(文字列)

VarType
変数の形式を表す整数型の値を返す。

VarType(変数名)

W

Weekday
何曜日であるかを表す整数を表すバリアント型の値を返す。

Weekday(日付，[週の第 1 日目の曜日])

WeekdayName
指定された曜日を表す文字列を返す。

WeekdayName(曜日を表す数値,曜日名の省略，曜日の最初の日)

Y

Year
年を表すバリアント型の値を返す。

Year

ASCII 文字コード一覧

Chr 関数などを使って文字を番号値で参照する時には ASCII コードを使用します。

番号	文字	番号	文字	番号	文字	番号	文字	番号	文字	番号	文字
0	□	40	(	80	P	120	x	160	[Space]	200	ﾈ
1	□	41	)	81	Q	121	y	161	｡	201	ﾉ
2	□	42	*	82	R	122	z	162	｢	202	ﾊ
3	□	43	+	83	S	123	{	163	｣	203	ﾋ
4	□	44	,	84	T	124			164	､	204	ﾌ
5	□	45	-	85	U	125	}	165	･	205	ﾍ
6	□	46	.	86	V	126	~	166	ｦ	206	ﾎ
7	□	47	/	87	W	127	□	167	ｧ	207	ﾏ
8	[BS]	48	0	88	X	128	□	168	ｨ	208	ﾐ
9	[TAB]	49	1	89	Y	129	□	169	ｩ	209	ﾑ
10	[LF]	50	2	90	Z	130	□	170	ｪ	210	ﾒ
11	□	51	3	91	[	131	□	171	ｫ	211	ﾓ
12	□	52	4	92	¥	132	□	172	ｬ	212	ﾔ
13	[CR]	53	5	93	]	133	□	173	ｭ	213	ﾕ
14	□	54	6	94	^	134	□	174	ｮ	214	ﾖ
15	□	55	7	95	_	135	□	175	ｯ	215	ﾗ
16	□	56	8	96	`	136	□	176	ｰ	216	ﾘ
17	□	57	9	97	a	137	□	177	ｱ	217	ﾙ
18	□	58	:	98	b	138	□	178	ｲ	218	ﾚ
19	□	59	;	99	c	139	□	179	ｳ	219	ﾛ
20	□	60	<	100	d	140	□	180	ｴ	220	ﾜ
21	□	61	=	101	e	141	□	181	ｵ	221	ﾝ
22	□	62	>	102	f	142	□	182	ｶ	222	ﾞ
23	□	63	?	103	g	143	□	183	ｷ	223	ﾟ
24	□	64	@	104	h	144	□	184	ｸ	224	□
25	□	65	A	105	i	145	□	185	ｹ	~以降 255 まで□	
26	□	66	B	106	j	146	□	186	ｺ		
27	□	67	C	107	k	147	□	187	ｻ		
28	□	68	D	108	l	148	□	188	ｼ		
29	□	69	E	109	m	149	□	189	ｽ		
30	□	70	F	110	n	150	□	190	ｾ		
31	□	71	G	111	o	151	□	191	ｿ		
32	[Space]	72	H	112	p	152	□	192	ﾀ		
33	!	73	I	113	q	153	□	193	ﾁ		
34	"	74	J	114	r	154	□	194	ﾂ		
35	#	75	K	115	s	155	□	195	ﾃ		
36	$	76	L	116	t	156	□	196	ﾄ		
37	%	77	M	117	u	157	□	197	ﾅ		
38	&	78	N	118	v	158	□	198	ﾆ		
39	'	79	O	119	w	159	□	199	ﾇ		

※□の文字は Windows では使用できない文字です。
※コード番号 8、9、10、13 は制御文字で、それぞれバックスペース、タブ、ラインフィード、キャリッジリターンです。

データ型一覧

変数を宣言するときにはデータ型を指定して、変数をどのような形式で格納するかを定めます。型によって扱える値の範囲やサイズは決まっています。Visual Basic のデータ型は以下のようになっています。

データ型	サイズ	範囲
バイト型(Byte)	1 バイト	0～255
ブール型(Boolean)	2 バイト	真(True)または偽(False)
整数型(Integer)	2 バイト	-32,768～32,767
長整数型(Long)	4 バイト	-2,147,483,648～2,147,483,647
単精度浮動小数点数型 (Single)	4 バイト	-3.402823E38～-1.401298E-45 1.401298E-45～3.402823E38
倍精度浮動小数点数型 (Double)	8 バイト	-1.79769313486232E308～ -4.94065645841247E-324 4.94065645841247E-324～ 1.79769313486232E308
通貨型(Currency)	8 バイト	-922,337,203,685,477.5808～ 922,337,203,685,477.5807
10 進型(Decimal)	14 バイト	(小数部分を持たない数値の場合) -79,228,162,514,264,337,593,543,950,335～ 79,228,162,514,264,337,593,543,950,335 (小数点以下 28 桁の数値の場合) -7.9228162514264337593543950335～ 7.9228162514264337593543950335 (絶対値の最小値は 0 を除いた場合) 0.0000000000000000000000000001
日付型(Date)	8 バイト	西暦 100 年 1 月 1 日～西暦 9999 年 12 月 31 日
オブジェクト型(Object)	4 バイト	オブジェクトを参照するデータ型
文字列型(String)(可変長)	10 バイト ＋文字列長	0～2GB
文字列型(固定長)	文字列長	1～2GB
バリアント型(Variant)(数値)	16 バイト	倍精度浮動小数点数型（Double）の範囲と同じ
バリアント型(Variant) (文字列)	22 バイト ＋文字列長	可変長文字列型（String）の範囲と同じ
ユーザー定義型	要素に依存	それぞれの要素の範囲はそのデータ型の範囲と同じ

※配列には、20 バイトのメモリと各配列の次元ごとに 4 バイト、そしてデータそのもののバイト数を合計したメモリが最低限必要です。
※配列を含むバリアント型には、配列だけのときより、12 バイト大きいメモリが必要です。

エラー番号一覧（一部）

トラップできる主なエラー番号とそのメッセージの一覧です。

番号	メッセージ
3	Returnに対応するGoSubがありません。
5	プロシージャの呼び出し、または引数が不正です。
6	オーバーフローしました。
7	メモリが足りません。
9	インデックスが有効範囲にありません。
10	この配列は固定されているか、または一時的にロックされています。
11	0で除算しました。
13	型が一致しません。
14	文字列領域が不足しています。
16	式が複雑すぎます。
17	要求された操作は実行できません。
18	ユーザーによる割り込みが発生しました。
20	エラーが発生していないときにResumeを実行することはできません。
28	スタック領域が不足しています。
35	Sub、Function、またはPropertyが定義されていません。
47	DLLのクライアントアプリケーションが多すぎます。
48	DLL読み込み時のエラーです。
49	DLLが正しく呼び出せません。
51	内部エラーです。
52	ファイル名または番号が不正です。
53	ファイルが見つかりません。
54	ファイルモードが不正です。
55	ファイルは既に開かれています。
57	デバイスI/Oエラーです。
58	既に同名のファイルが存在しています。
59	レコード長が一致しません。
61	ディスクの空き容量が不足しています。
62	ファイルにこれ以上データがありません。
63	レコード番号が不正です。
67	ファイルが多すぎます。
68	デバイスが準備されていません。
70	書き込みできません。
71	ディスクが準備されていません。
74	ディスク名は変更できません。
75	パス名が無効です。
76	パスが見つかりません。
91	オブジェクト変数またはWithブロック変数が設定されていません。
92	Forループが初期化されていません。
93	パターン文字列が不正です。
94	Nullの使い方が不正です。
97	オブジェクトが定義クラスのインスタンスではない場合、このオブジェクトに関するフレンド関数は呼び出せません。
298	システムDLLをロードできません。
320	キャラクタデバイスは使えません。
321	不正なファイル形式です。
322	必要なテンポラリファイルを作成できません。
325	リソースファイルの形式が不正です。
327	データ値が見つかりません。
328	不正なパラメータです。配列に書き込めません。
335	システムレジストリにアクセスできません。
336	ActiveXコンポーネントが正しく登録されていません。
337	ActiveXコンポーネントが見つかりません。
338	ActiveXコンポーネントが正常に実行されませんでした。
360	このオブジェクトは既にロードされています。
361	このオブジェクトは、ロードまたはアンロードすることはできません。
363	指定されたActiveXコントロールが見つかりません。
364	既にアンロードされています。
365	現在アンロードできません。
368	ファイルは古い形式で作成されています。このプログラムには新しい形式のファイルが必要です。
371	指定されたオブジェクトは、Showメソッドのオーナーフォームとして使用できません。
380	プロパティの値が不正です。
381	プロパティ配列のインデックスが不正です。

番号	メッセージ
382	プロパティは、実行時には設定できません。
383	プロパティは値の取得のみ可能です。
385	このプロパティには配列のインデックスが必要です。
387	プロパティは値を設定できません。
393	プロパティは実行時に値の取得はできません。
394	プロパティは設定のみ可能です。
400	既にフォームは表示されています。モーダルにできません。
402	一番手前（前面）のモーダルフォームを先に閉じてください。
419	オブジェクトを利用できません。
422	プロパティが見つかりません。
423	プロパティまたはメソッドが見つかりません。
424	オブジェクトが必要です。
425	オブジェクトの使い方が不正です
429	ActiveX コンポーネントはオブジェクトを作成できません。
430	クラスはオートメーションをサポートしていません。
432	オートメーションの操作中にファイル名またはクラス名を見つけられませんでした。
438	オブジェクトは、このプロパティまたはメソッドをサポートしていません。
440	オートメーションエラーです。
442	リモートプロセス用のタイプライブラリまたはオブジェクトライブラリへの参照は失われました。
443	オートメーションオブジェクトには既定値がありません。
445	オブジェクトはこの動作をサポートしていません。
446	オブジェクトは名前付き引数をサポートしていません。
447	オブジェクトは現在の国別情報の設定をサポートしていません。
448	名前付き引数が見つかりません。
449	引数は省略できません。または不正なプロパティを指定しています。
450	引数の数が一致していません。または不正なプロパティを指定しています。
451	このオブジェクトがコレクションではありません。
452	序数が不正です。
453	関数は指定された DLL には定義されていません。
454	コードリソースが見つかりません。
455	コードリソースのロックエラー
457	このキーは既にこのコレクションの要素に割り当てられています。
458	Visual Basic でサポートされていないオートメーションが変数で使用されています。
459	このコンポーネントでは、イベントはサポートされていません。
460	クリップボードのデータ形式が不正です。
461	データの形式が一致しません。
480	AutoRedraw イメージを作成できません。
481	ピクチャが不正です。
482	プリンタエラーです。
483	プリンタドライバは指定されたプロパティをサポートしていません。
484	システムからプリンタ情報を受けるときに問題が発生しました。プリンタが正しく設定されているかを確かめてください。
485	ピクチャの形式が不正です。
486	フォームのイメージをこのプリンターで印刷することはできません。
520	クリップボードを空にできません。
521	クリップボードを開けません。
735	テンポラリファイルに保存できません。
744	検索文字列が見つかりませんでした。
746	置換後の文字列が長すぎます。
31001	メモリが不足しています。
31004	オブジェクトがありません。
31018	クラスが設定されていません。
31027	オブジェクトをアクティブにできません。
31032	埋め込みオブジェクトが作成できません。
31036	ファイルへの書き込み中にエラーが発生しました。
31037	ファイルの読み込み中にエラーが発生しました。

プロジェクトファイルの形式

Visual Basicでは、コントロールを配置したり、プロジェクトファイルを実行した時などにいくつかのファイルが作成されます。プロジェクトやプログラムによって必要なファイルは異なります。

●デザイン時に作られるファイルおよびその他のファイル

拡張子	説明
.bas	標準モジュール
.cls	クラスモジュール
.ctl	ユーザーコントロールファイル
.ctx	ユーザーコントロールバイナリファイル
.dca	Activeデザイナキャッシュ
.ddf	ディストリビューションウィザードのCAB情報ファイル
.dep	セットアップウィザードの依存ファイル
.dob	ActiveXドキュメントフォームファイル
.dox	ActiveXドキュメントバイナリフォームファイル
.dsr	Activeデザイナファイル
.dsx	Activeデザイナバイナリファイル
.dws	ディストリビューションウィザードのスクリプトファイル
.frm	フォームモジュールファイル
.frx	バイナリフォームファイル

拡張子	説明
.log	読み込みエラー時に生成されるログ
.oca	コントロールタイプライブラリキャッシュファイル
.pag	プロパティページファイル
.pgx	バイナリプロパティページファイル
.res	リソースファイル
.tlb	リモートオートメーションタイプライブラリファイル
.vbg	Visual Basicグループプロジェクトファイル
.vbl	コントロールライセンスファイル
.vbp	Visual Basicプロジェクトファイル
.vbr	リモートオートメーション登録ファイル
.vbw	Visual Basicプロジェクトワークスペースファイル
.vbz	ウィザード起動ファイル
.wct	WebClass HTMLテンプレート

●実行時に生成されるファイル

拡張子	説明
.dll	インプロセスActiveXコンポーネント
.exe	実行可能ファイルまたはActiveXコンポーネント
.ocx	ActiveXコントロール

拡張子	説明
.vbd	ActiveXドキュメントファイル
.wct	WebClass HTMLテンプレート

ショートカット一覧

ショートカットを使うと作業を効率的化できます。

●デザイン時のショートカット

キー	説明
Alt+F4	タスクの終了
Ctrl+→	コントロールを右移動
Ctrl+←	コントロールを左移動
Ctrl+↑	コントロールを上移動
Ctrl+↓	コントロールを下移動

キー	説明
Ctrl+Break	実行を停止
Ctrl+C	コピー
Ctrl+E	メニューエディタ
Ctrl+F4	ウィンドウを閉じる
Ctrl+F6	ウィンドウの切り替え

キー	説明	キー	説明
Ctrl+G	イミディエイトウィンドウ	F4	プロパティウィンドウ
Ctrl+J	前面へ移動	F5	実行／再開
Ctrl+K	後ろへ移動	F7	コードウィンドウ表示
Ctrl+Tab	ウィンドウの切り替え	F8	1行ずつコードを実行
Ctrl+V	貼り付け	Home	最初のプロパティに移動
Ctrl+W	ウォッチ式の追加	PageDown	次のページを表示
Ctrl+X	切り取り	PageUp	前のページを表示
Ctrl+Z	元に戻す	Shift+F10	ショートカットを表示
Ctrl+クリック	複数のコントロールの選択	Shift+F5	始めから再実行
Delete	削除	Shift+F8	プロシージャごとに実行
End	最後のプロパティに移動	Shift+Tab	コントロール間を移動
Esc	キャンセル	Shift+クリック	複数のコントロールの選択
F1	ヘルプを表示	Tab	コントロール間を移動
F2	オブジェクトブラウザ	ダブルクリック	プロパティの値の切り替え

●コード作成時のショートカット

キー	説明	キー	説明
Ctrl+→	右の単語に移動	Ctrl+Shift+F9	すべてのブレークポイントをクリア
Ctrl+←	左の単語に移動	Ctrl+Shift+I	[パラメータヒント]
Ctrl+↑	前のプロシージャ	Ctrl+Shift+J	[定数の一覧]を表示
Ctrl+↓	次のプロシージャ	Ctrl+Space	[入力候補]を表示
Ctrl+BackSpace	単語の先頭までを削除	Ctrl+V	貼り付け
Ctrl+Break	中断	Ctrl+X	切り取り
Ctrl+C	コピー	Ctrl+Y	現在行を切り取り
Ctrl+D	ファイルの追加	Ctrl+Z	元に戻す
Ctrl+Delete	単語の最後までを削除	Delete	削除
Ctrl+End	コードの最後に移動	Delete	削除
Ctrl+F	検索	End	行末に移動
Ctrl+F2	オブジェクトに移動	F3	次を検索
Ctrl+F5	コンパイル後に開始	F4	プロパティウィンドウ
Ctrl+F8	カーソルの前まで実行	F6	分割コードウィンドウ間を移動
Ctrl+G	イミディエイトウィンドウに移動	F8	シングルステップ実行
Ctrl+H	置換	F9	ブレークポイントの設定/解除
Ctrl+Home	コードの先頭に移動	Home	行頭に移動
Ctrl+I	[クイックヒント]	Shift+F2	定義の表示
Ctrl+J	[プロパティ/メソッドの一覧]	Shift+F3	前を検索
Ctrl+N	新しいプロジェクト	Shift+F4	プロパティページ
Ctrl+O	プロジェクトを開く	Shift+F5	再実行
Ctrl+P	印刷	Shift+F8	プロシージャステップ
Ctrl+PageDown	プロシージャの最後に移動	Shift+F9	クイックウォッチ
Ctrl+PageUp	プロシージャの先頭に移動	Shift+F10	ショートカットの表示
Ctrl+R	プロジェクトエクスプローラ	Shift+Tab	インデントを戻す
Ctrl+S	フォームの上書き保存	Tab	インデント

●メニュー作成時のショートカット

キー	説明
Alt+↑	リストの開閉
Alt+↓	リストの開閉
Alt+B	メニュー項目の階層レベルを維持したまま下に移動
Alt+L	メニュー項目をメニュー階層の上位レベルに移動
Alt+R	メニュー項目をメニュー階層の下位レベルに移動
Alt+U	メニュー項目の階層レベルを維持したまま上に移動
End	リスト内の最後の項目に移動
Enter	メニュー項目を前方に向かって順に移動
F4	リストの開閉
Home	リスト内の最初の項目に移動

●ウォッチウィンドウのショートカット

キー	説明
Ctrl+W	［ウォッチ式の追加］ダイアログボックスを表示
Enter	選択されたウォッチ値の左側にプラス記号（+）またはマイナス記号（-）があるときに、展開または折りたたみを行う
F2	オブジェクトブラウザを表示
F6	ウォッチウィンドウとイミディエイトウィンドウを切り替え
Shift+Enter	選択されたウォッチ式を表示
Shift+F10	ショートカットメニューを表示

●イミディエイトウィンドウのキーボードショートカット

キー	説明
Alt+F5	エラー処理ルーチンを実行するか、呼び出し側のプロシージャにエラーを返す
Alt+F8	エラー処理ルーチンをシングルステップで実行するか、呼び出し側のプロシージャにエラーを返す
Ctrl+End	イミディエイトウィンドウの最後にカーソルを移動
Ctrl+Enter	キャリッジリターンを挿入
Ctrl+Home	イミディエイトウィンドウの先頭にカーソルを移動
Ctrl+L	［呼び出し履歴］ダイアログボックスを表示（中断モード時のみ）
Del	選択された文字列を削除
Enter	選択されたコード行を実行

A 練習問題の解答

Lesson1・・・・・・A1 (Q&A-01.vbp)

メッセージボックスの文字を変えるには、MsgBoxの後ろの文字を変更します。表示する文字を必ず "（ダブルクォーテーション）で囲みます。

```
Private Sub Command1_Click()
    MsgBox "こんにちは、みなさん！"
End Sub
```

Lesson2・・・・・・A1 (Q&A-02.vbp)

コマンドボタンの表面の文字は、［Caption］プロパティで設定します。

［終了］ボタンをクリックして、プロパティウィンドウで［Caption］プロパティに設定されている文字を変更します。

ここを「閉じる」に変更

Lesson3・・・・・・A1 (Q&A-03.vbp)

ラベルコントロールの背景色は、［BackColor］プロパティで設定します。

ここをクリック

一覧から黄緑色を選択

Lesson3・・・・・・A2 (Q&A-03.vbp)

フォントの種類は［Font］プロパティで指定します。

Lesson4・・・・・・A1 (Q&A-04.vbp)

コマンドボタンがクリックされたときに、ラベルコントロールの背景色を変更するので、コマンドボタンのClickイベントプロシージャに処理を作成します。

```
Private Sub Blue_Click()
    Label1.BackColor = &HC00000
End Sub
```

Lesson4・・・・・・A2 (Q&A-04.vbp)

ラベルコントロールの文字の色を変更するには、ラベルコントロールの［ForeColor］プロパティで行います。

```
Private Sub Pink_Click()
    Label1.ForeColor = &HFF80FF
End Sub
```

Lesson5・・・・・・A1 (Q&A-05.vbp)

イメージコントロールに画像を設定するには、［Picture］プロパティに使用する画像のファイルを設定します。

Lesson6・・・・・・A1 (Q&A-06.vbp)

必ず、ファイル名をフルパスで指定してください。以下は、ドライブCにある「Free.bmp」という名前のファイルを指定しています。

```
Private Sub Image1_Click()
    'パス名は自分のコンピュータにあわせてください
    Image1.Picture = LoadPicture("c:¥Free.bmp")
End Sub
```

Lesson7・・・・・・A1 (Q&A-07.vbp)

いずれも、背景色の設定は［BackColor］プロパティで行います。必ず変更するコントロールを選択状態にしてから、プロパティを設定します。ピンクは、「&H008080FF&」です。

Lesson8・・・・・・A1 (Q&A-08.vbp)

フォームがダブルクリックされると、「DblClick」というイベントが発生しますので、このイベントプロシージャ「Form_DblClick()」に、フォームの背景色を変更する処理を記述します。

```
Private Sub Form_DblClick()
    Form1.BackColor = &HFFFF&       追加したコード
End Sub

Private Sub Image1_Click()
    Image1.BorderStyle = 1
End Sub

Private Sub Image1_DblClick()
    MsgBox "ダブルクリックされました"
    Image1.BorderStyle = 0
End Sub
```

Lesson9・・・・・・A1 (Q&A-09.vbp)

MouseDownイベントプロシージャでは、マウスの右ボタンが押されると、変数Buttonには「2」が格納されます。Ifステートメントで、クリックされたボタンが右ボタン（2）かどうかをチェックし、右ボタンであればラベルコントロールの［Left］［Top］プロパティに、マウスの座標「X」と「Y」を代入します。

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, _
                           X As Single, Y As Single)
    If Button = 2 Then
        Label1.Left = X
        Label1.Top = Y
    End If
End Sub
```

Lesson10 ・・・・A1 (Q&A-10.vbp)

メッセージボックスを使ってマウスの座標位置を表示するには、マウスの座標位置「X」と「Y」を、文字列連結演算子「&」を使って文字列とつなげます。

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, _
                         X As Single, Y As Single)
    Dim Message As String

    If Button = 1 Then
        Message = "マウスの位置は" & X & "," & Y
        MsgBox Message
    End If
End Sub
```

Lesson11 ・・・・A1 (Q&A-11.vbp)

線の種類を変更するには、［DrawStyle］プロパティを使用します。これを、コマンドボタンのClickイベントプロシージャで行います。

```
Dim StartX As Integer
Dim StartY As Integer

Private Sub Command1_Click()
    DrawStyle = 2
End Sub
```

追加したコード

Lesson12 ・・・・A1 (Q&A-12.vbp)

水色のカラーパレットを追加するには、ラベルコントロールを追加し、Clickイベントプロシージャで変数CColorに「11」をセットします。また、線の太さは、

[DrawWidth] プロパティの値を「7」に設定します。

```
Private Sub Label6_Click()
    CColor = 11
End Sub

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, _
                          X As Single, Y As Single)
    Form1.DrawWidth = 7
    If Button = 1 Then
        PSet (X, Y), QBColor(CColor)
    End If
End Sub
```

追加したコード

変更

Lesson12 ････A2 (Q&A-122.vbp)

イメージコントロールをフォームに１つ配置し、[Picture] プロパティに「Erase01.ico」を設定します。このイメージのClickイベントプロシージャで、描画の処理を作成しますが、このとき、色の設定に「7」を指定し、フォームの背景色と同色で描画するようにします。

```
Private Sub Image1_Click()
    CColor = 7
End Sub
```

Lesson13 ････A1 (Q&A-13.vbp)

メニューエディタを使って、メニューを追加する練習です。

メニュー エディタ

キャプション(P): 貼りつけ

名前(M): IDMPaste

インデックス(X):　ショートカット(S): (なし)

ヘルプ コンテキスト番号(H): 0　ネゴシエート位置(O): 0 - なし

チェック(C)　有効(E)　表示(V)　ウィンドウ リスト(W)

OK　キャンセル　次へ(N)　挿入(I)　削除(T)

ファイル
････新規作成
････-
････終了(&x)
編集
････切り取り
････コピー(&c)
････-
････貼りつけ
････････貼りつけ
････････ファイルから

Lesson14 ････A1 (Q&A-14.vbp)

ポップアップメニューは、マウスの右ボタンが押されたときに表示されます。処理は、フォームのMouseDownイベントプロシージャを使います。

まず、Ifステートメントで押されたボタンが右ボタンかどうかを判断します。そして、右ボタンであればPopupMenuメソッドで、[編集] メニューの [名前] で入力したオブジェクト名を指定します。

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, _
                           X As Single, Y As Single)
    If Button = 2 Then
        PopupMenu IDMEdit, vbPopupMenuRightButton
    End If
End Sub
```

Lesson15 ････A1 (Q&A-15.vbp)

メニューのキャプションは、[Caption] プロパティに文字列を直接指定します。また、Ifステートメントでメニューがいくつ追加されたのかをチェックしておきます。そして、2つめが追加された時点で、コマンドボタンの [Enabled] プロパティをFalseに設定します。これで、コマンドボタンはもうクリックすることができなくなるので、メニューを追加できません。

メニューを追加するLoadステートメントは、それぞれのIf文の中で実行しています。

```
Private Sub Command1_Click()
    iCount = iCount + 1
    If iCount = 1 Then
        Load Separator(iCount)
        Separator(1).Caption = "追加"
    Else If iCount = 2 Then
        Load Separator(iCount)
        Separator(2).Caption = "印刷"
        Command1.Enabled = False
    End If
End Sub
```

Lesson16 ････A1 (Q&A-16.vbp)

追加されたメニューが選択された場合は、メニューのSeparatorのClickイベントプロシージャが呼び出されます。そして、そのイベントプロシージャの引数「Index」に、何番目に追加されたメニューが選択されたのか、その番号が格納されています。そこで、Indexの値を調べて、Ifステートメントでラベルコントロールの色を変える処理を実行します。

```
Private Sub separator_Click(Index As Integer)
    Dim Msg As String

    If Index = 2 Then
        Label1.ForeColor = &HFF&
    Else
        Label1.ForeColor = &H0
    End If
    Msg = Separator(Index).Caption & "が選択されました"
    Label1.Caption = Msg
End Sub
```

追加したコード

Lesson17 ････A1 (Q&A-17.vbp)

メニューエディタで、［このプログラムについて］メニューを追加し、メニューの「名前」を「IDMAbout」にします。そして、このメニューのClickイベントプロシージャで、Form2をShowメソッドを使って表示します。

```
Private Sub IDMAbout_Click()
    Form2.Show 1
End Sub
```

Lesson17 ････A2 (Q&A-17.vbp)

フォームを消去するには、Unloadステートメントを使います。

```
Private Sub Command1_Click()
    Unload Form2
End Sub
```

Lesson18 ････A1 (Q&A-18.vbp)

①フォームに、ラベルコントロールを配置し、［Caption］プロパティに「Welcome」という文字を設定します。
②タイマーコントロールをフォームに配置します。
③フォームのLoadイベントプロシージャで、タイマーコントロールの［Interval］プロパティに「100」を設定します。また、変数xに「5000」を代入します。
④文字を動かすには、タイマーコントロールのTimerイベントプロシージャで、ラベルコントロールの［Left］プロパティの値を変えることで座標位置を変更していきます。

```
Dim x As Integer

Private Sub Form_Load()
    Timer1.Interval = 100
    x = 5000
```

```
End Sub
Private Sub Timer1_Timer()
    If x = -1000 Then
        x = 5000
    End If
    x = x - 50
    Label1.Left = x
End Sub
```

Lesson19 ････A1 (Q&A-19.vbp)

①フォームに、スクロールバーコントロールとテキストボックスコントロールを配置します。

②スクロールバーコントロールは、次のプロパティを設定します。

プロパティ	値
Max	1000
Min	1
LargeChange	10
SmallChange	1

③スクロールバーコントロールのChangeイベントプロシージャで、タイマーコントロールの［Interval］プロパティに、スクロールバーコントロールの［Value］プロパティの値を代入します。また、テキストボックスコントロールの［Text］プロパティにも値を代入します。

④テキストボックスコントロールでは、入力された数値をスクロールバーコントロールの［Value］プロパティに代入します。

⑤スクロールバーコントロールの［Value］プロパティに値が代入されると、スクロールバーコントロールに自動的にChangeイベントが発生します。そして、Changeイベントプロシージャが実行され、タイマーコントロールの［Interval］プロパティがテキストボックスコントロールに入力された値で設定されます。

```
Dim x As Integer

Private Sub Form_Load()
    Timer1.Interval = 100
    x = 5000
End Sub

Private Sub HScroll1_Change()
    Text1.Text = HScroll1.Value
    Timer1.Interval = HScroll1.Value
End Sub
```

追加したコード

```
Private Sub Text1_Change()
    If Text1.Text = "" Or Text1.Text = "0" Then
        MsgBox "数値を入力してください"
        HScroll1.Value = 1
        Text1.Text = "1"
    Else
        HScroll1.Value = Int(Text1.Text)
    End If
End Sub

Private Sub Timer1_Timer()
    If x = -1000 Then
        x = 5000
    End If
    x = x - 50
    Label1.Left = x
End Sub
```

追加したコード

Lesson20 ••••A1 (Q&A-20.vbp)

チェックボックスがチェックされたかどうかは、チェックボックスコントロールの［Value］プロパティの値を調べます。チェックされていれば、ラベルコントロールの［FontItalic］プロパティをTrueに設定します。チェックが外されれば、［FontItalic］プロパティをFalseに設定します。

```
Private Sub Check2_Click()
    Select Case Check2.Value
        Case 1
            Label2.FontItalic = True

        Case 0
            Label2.FontItalic = False
     End Select
End Sub
```

追加したコード

Lesson21 ••••A1 (Q&A-21.vbp)

［削除］ボタンの無効化は、［削除］ボタンのClickイベントプロシージャの中で行います。常に、コンボボックスのListCountプロパティの値をチェックして、この値が「0」になった時点でボタンの無効処理を行うようにします。

```
Private Sub DelBtn_Click()
    Dim Result As Integer
    Dim Name As String
    Dim Msg As String
```

```
        Name = Combo1.Text
        Msg = "リストボックスから" & Name & " を削除しますか"
        Result = MsgBox(Msg, vbYesNo)
        If Result = vbYes Then
            Combo1.RemoveItem Combo1.ListIndex
            If Combo1.ListCount = 0 Then
                MsgBox "これ以上削除できません"
                DelBtn.Enabled = False
            End If
         End If
    End Sub
```

追加したコード

また、［削除］ボタンの有効化は、［追加］ボタンのClickイベントプロシージャで行います。ここでも、毎回コンボボックスの［ListCount］プロパティの値を調べて、「0」以外であれば常にボタンが有効になるように処理しておきます。

```
Private Sub AddBtn_Click()
    Dim Result As Integer
    Dim Name As String

    Name = Combo1.Text
    Result = MsgBox("リストボックスに追加しますか？", vbYesNo)
    If Result = vbYes Then
        Combo1.AddItem Name
        If Combo1.ListCount <> 0 Then
            DelBtn.Enabled = True
        End If
    End If
End Sub
```

追加したコード

Lesson 22・・・・A1 (Q&A-22.vbp)

Line Input#ステートメントの記述を、「Input #1」に変更します。

```
Private Sub FileRead()
    Dim Buff1 As String
    Dim Buff2 As String

    Open App.Path & "¥readme.txt" For Input As #1
    Do While Not EOF(1)
        Input #1, Buff1      ← 変更
        Buff1 = Buff1 + Chr(13) + Chr(10)
        Buff2 = Buff2 + Buff1
    Loop
```

```
        Text1.Text = Buff2
        Close #1
End Sub
```

Lesson 23・・・・A1 (Q&A-23.vbp)

①メッセージボックスは、引数に「vbYesNoCancel」を指定し、［はい］［いいえ］［キャンセル］ボタンの付いたタイプを作成します。
②MsgBox関数が実行されると、クリックされたボタンのタイプが戻り値として返ってきますので、これを変数で受け止めます。
③戻り値は定数で設定されています。［はい］がクリックされると「vbYes」が返されますから、このときだけリストボックスの内容を保存するプロシージャ「FileSave」を呼び出します。そして、Endステートメントでプログラムを終了させます。
④［いいえ］ボタンがクリックされると、MsgBox関数は「vbNo」を返してきます。このときは、リストボックスの項目は保存せず、Endステートメントでプログラムを終了するだけです。
⑤［キャンセル］ボタンがクリックされたときは、メッセージボックスをキャンセルするだけですから、特に何も記述しません。
⑥Form_Unloadイベントプロシージャで、FileSaveを呼び出していましたが、メッセージボックスで［はい］がクリックされたときだけ保存しますので、このプロシージャ全体を削除します。

```
Private Sub ExitBtn_Click()
    Dim Result As Integer

    Result = MsgBox("リスト項目を保存しますか", vbYesNoCancel, _
                "データの保存")
    If Result = vbYes Then
        Call FileSave
        End
    ElseIf Result = vbNo Then
        End
    End If
End Sub
```

Lesson 24・・・・A1 (Q&A-24.vbp)

①フォームに、コンボボックスを追加します。
②コンボボックスの［List］プロパティに、「*.bmp」「*.jpg」「*.gif」の文字列を入力します。
③ファイルリストボックスの［Pattern］プロパティを「*.bmp」1つだけにしておきます。

④次のコードを、コンボボックス（Combo1）のClickイベントプロシージャに追加します。これで、コンボボックスで選択したファイルタイプが、ファイルリストボックスの［Pattern］プロパティに代入されます。しかし、ただ代入しただけでは、ファイルリストボックスは変更されませんので、Refreshメソッドを使って、強制的に表示を更新します。

```
Private Sub Combo1_Click()
    File1.Pattern = Combo1.Text
    File1.Refresh
End Sub
```

Lesson 25・・・・A1 (Q&A-25.vbp)

プログラム終了時に、バージョン番号を保存するので、［終了］ボタンがクリックされたときに、処理を実行するようにします。

まず、プログラムのバージョン番号は、Appオブジェクトの［Major］［Minor］［Revisoin］プロパティに格納されています。これらの値を組み合わせて、バージョン番号になるように、文字列を作成します。このとき、同じオブジェクト名「App」を何回も使いますので、Withステートメントを使って、オブジェクト名の記述を一回で済ませています。

バージョン番号が作成できたら、SaveSettingステートメントを使ってレジストリに保存します。新しく「Version」というsectionが設定されています。確かめてみてください。

```
Private Sub Exit_Click()
    Dim VerNum As String

    With App
        VerNum = "Version : " & .Major & "/ " & .Minor _
                 & "/ " & .Revision
    End With
    SaveSetting appname:="QRun", section:="Version", _
              Key:="Build", setting:=VerNum
    End
End Sub
```

Lesson 26・・・・A1 (Q&A-26.vbp)

①フォーム に、コモンダイアログボックスコントロール、コマンドボタン、テキストボックスコントロールを配置します。

Form1

ファイルを開く　　閉じる

Text1

②テキストボックスコントロールの［Multiline］プロパティを「True」に、［ScrollBars］プロパティを「3－両方」に設定します。
③コモンダイアログボックスに、テキストファイルだけを表示するために、［Filter］プロパティを「テキストファイル(*.txt)¦*.txt」に設定します。

プロパティ－ CommonDialog1

CommonDialog1 CommonDialog

全体　項目別

DefaultExt	
DialogTitle	
FileName	
Filter	テキストファイル(*.txt)¦*.txt
FilterIndex	0
Flags	0
FontBold	False
FontItalic	False

④［ファイルを開く］ボタンにコードを記述します。コモンダイアログボックスの［FileName］プロパティに格納されるファイル名を、変数GetNameに格納し、これをOpenステートメントに渡してファイルを開きます。
⑤［ファイルを開く］ダイアログボックスで［キャンセル］ボタンが押されるか、ファイル名が空のときには処理を行わないようにします。
⑥後は、ファイルを読み込む操作を行って、テキストボックスコントロールの［Text］プロパティに出力します。
⑦［閉じる］ボタンにEndステートメントを記述します。

```
Private Sub Command1_Click()
    Dim GetName As String
    Dim Buff1 As String
    Dim Buff2 As String
```

```
    With CommonDialog1
        .ShowOpen
        GetName = .filename
    End With

    If GetName <> "" Then
        Open GetName For Input As #1
        Do While Not EOF(1)
            Line Input #1, Buff1
            Buff1 = Buff1 + Chr(13) + Chr(10)
            Buff2 = Buff2 + Buff1
        Loop
        Text1.Text = Buff2
        Close #1
    EndIf
End Sub

Private Sub Command2_Click()
    End
End Sub
```

Lesson 27・・・・A1 (Q&A-27.vbp)

MIDIファイルを再生するには、まず［DeviceType］プロパティに「Sequencer」をセットします。そして、［filename］プロパティにMIDIファイルを設定します。

```
Private Sub Form_Load()
    MMControl1.DeviceType = "Sequencer"
    MMControl1.filename = App.Path & "¥Md001.mid"
    MMControl1.Notify = False
    MMControl1.Wait = True
    MMControl1.Shareable = False

    MMControl1.Command = "Open"
End Sub

Private Sub Form_Unload(Cancel As Integer)
    MMControl1.Command = "Close"
End Sub
```

※この練習問題で使用した「Md001.mid」の著作権は（株）C&R研究所が所有します。

Lesson 28・・・・A1 (Q&A-28.vbp)

①「Lesson28.vbp」のプログラムに、コモンダイアログボックスコントロールを追加します。

②フォームのLoadイベントプロシージャから、次の行を削除します。

```
MMControl1.filename = App.Path & "¥music.wav"
```

③コマンドボタンのClickイベントプロシージャに、コモンダイアログボックスの処理を追加します。

```
With CommonDialog1
    .Filter = "サウンドファイル (*.wav)|*.wav"
    .ShowOpen
    GetName = .filename
End With
```

④選択されたファイル名を、コモンダイアログボックスコントロールの［filename］プロパティから取得し、マルチメディアコントロールの［filename］プロパティに設定します。

```
MMControl1.filename = GetName
```

⑤あとは、For...Nextループでサウンドを再生します。

```
Private Sub Command1_Click()
    Dim i As Integer
    Dim GetName As String

    With CommonDialog1
        .Filter = "サウンドファイル (*.wav)|*.wav"
        .ShowOpen
        GetName = .filename
    End With

    If GetName <> "" then
        MMControl1.filename = GetName

        For i = 1 To 3
            MMControl1.Command = "Sound"
        Next
    EndIf
End Sub

Private Sub Form_Load()
    MMControl1.DeviceType = "WaveAudio"
    MMControl1.Notify = False
    MMControl1.Wait = True
```

```
    MMControl1.Shareable = False

    MMControl1.Command = "Open"
End Sub

Private Sub Form_Unload(Cancel As Integer)
    MMControl1.Command = "Close"
End Sub
```

Lesson 29・・・・A1 (Q&A-29.vbp)

フォーム起動時に、[DatabaseName] プロパティに読み込むデータベースファイル名を指定します。このときに、App オブジェクトの Path プロパティを使って、実行するプログラムがあるディレクトリパスを取得します。ただし、このときファイル名の前に「¥」記号を付けてください。これを付け忘れると、ファイルは読み込まれません。

```
Private Sub Form_Load()
    Data1.DatabaseName = App.Path & "¥本社合計1.MDB"
End Sub
```

Lesson 30・・・・A1 (Q&A-30.vbp)

フォームにタブ付きダイアログボックスを配置します。
[Tabs] プロパティを「6」に設定し、[TabsPerRow] を「3」にセットします。

Lesson 31・・・・A1 (Q&A-31.vbp)

①Visual Basic の [プロジェクト] メニューから [フォームモジュールの追加] を選びます。フォーム選択ウィンドウが表示されますから、[バージョン情報] を選びます。

②「バージョン情報」フォームの、「バージョン」「アプリケーションの説明」「警告」を書き換えます。必要ならレイアウトも自由に変更してみてください。アイコンはVisual Basicのものが使われているので、自分の好きなイメージに入れ替えることもできます。

③このフォームも、「アプリケーションタイトル」「バージョン情報」は、コードで制御されています。また、「システム情報」は、「MSINFO32.EXE」というプログラムを呼び出しています。このプログラムがコンピュータにない場合は、メッセージボックスが表示されます。

④フォームが完成したら、Form1に配置したコマンドボタンのClickイベントプロシージャの処理を、バージョン情報フォームを表示する処理に変更します。

```
Private Sub Command1_Click()
    frmAbout.Show    ← 変更
End Sub
```

⑤これで、完成です。プログラムを実行して動作を確認してください。

Lesson 32････A1 (Q&A-32.vbp)

①まず、Visual Basicの［デバッグ］メニューから［ウォッチ式の追加］を選びます。
②ウィンドウが開きますので、［式］ボックスに変数名「i」と入力します。対象とするプロシージャは［Command1-Click］、モジュールは［Form1］を選びます。そして、［OK］ボタンをクリックします。

ウォッチ式の追加
式(E):
i
対象
ﾌﾟﾛｼｰｼﾞｬ(P): Command1_Click
ﾓｼﾞｭｰﾙ(M): Form1
ﾌﾟﾛｼﾞｪｸﾄ: Project1
ウォッチの種類
式のウォッチ(W)
式が True のときに中断(T)
式の内容が変化したときに中断(C)
OK
ｷｬﾝｾﾙ
ﾍﾙﾌﾟ(H)

③プログラムをステップ実行し、変数の値をウォッチウィンドウで確認します。

ウォッチ

式	値	型	対象
i	3	Integer	Form1.Command1_Click
x	2900	Integer	Form1.Command1_Click

Lesson 34････A1 (Q&A-34.vbp)

エラーが発生したときのエラー番号は、Errオブジェクトの［Number］プロパティに格納されます。これを、メッセージボックスで表示させます。

```
MSG:
    MsgBox Err.Number  ←――追加
    MsgBox "指定したファイルが見つかりません。再度入力して下さい。"
    Text1.Text = ""
End Sub
```

この例題では、MSGラベルのすぐ後に記述しています。エラー番号の意味については、付録の「エラー番号一覧」を参照してください。

索引

A

ActiveXコントロール xi
　HTMLへの組み込み 380
　Webでの公開 385
　外部ファイルの作成 379
　概要 362
　組み込み 375
　作成 364
　動作テスト 375
　配布 302
　フォームへの配置 379
AddItemメソッド 191
[Alignment] プロパティ 27
API（Application
　Programming Interface） x, xiv
APIビューア xxv
[App.Path] プロパティ 154
Appオブジェクト
　バージョン情報のプロパティ 347
　バージョン番号のプロパティ 347
ASCII文字コード一覧 408
[AutoRedraw] プロパティ 83

B

[BackColor] プロパティ 24
[BackStyle] プロパティ 53
Basic（Beginner's All-purpose
　Symbolic Instruction Code） x
[BorderStyle] プロパティ 59, 71

C

Cabファイル 357
Callステートメント 208
[Caption] プロパティ 6
Changeイベント 167
ChDirステートメント 230
[CheckBox] ボタン 176
Clickイベント 56
Closeステートメント 208, 214
Clsメソッド 93
[Color] プロパティ 280
[ComboBox] ボタン 187
[CommandButton] ボタン 5
[CommonDialog] ボタン 273

D

[DatabaseName] プロパティ 293
[DataField] プロパティ 296
[DataSource] プロパティ 296
[Data] ボタン 293
DblClickイベント 56
Debugオブジェクト
　Printメソッド 327
DeleteSettingステートメント 250
[DeviceType] プロパティ 285
DHTML（Dynamic HTML） xxvii
Dimステートメント 75
[DirListBox] ボタン 222
Do While...Loopステートメント 206
[DrawStyle] プロパティ 80
[DrawWidth] プロシージャ 91
[DriveListBox] ボタン 221
[Drive] プロパティ 223

E

[EjectVisible] プロパティ 284
Elseステートメント 122
[Enabled] プロパティ 109, 183
Endステートメント 132
EOF関数 206
Errオブジェクト 270
　[Number] プロパティ 335
EXE（実行ファイル） 3
　作成 338
　保存 340

F

FileDateTime関数 214
FileLen関数 214, 332
[FileListBox] ボタン 222
[Filename] プロパティ 278
[Filter] プロパティ 278
[Flags] プロパティ 279
[FontBold] プロパティ 279
[FontItalic] プロパティ 185, 279
[FontName] プロパティ 279
[FontSize] プロパティ 279
[FontStrikethru] プロパティ 279
[FontUnderline] プロパティ 279
[Font] プロパティ 26
[ForeColor] プロパティ 25, 280
[Frame] ボタン 47
FreeFile関数 214
.frm 18
FrontPage Express
　起動 381

G

GetSetting関数 246
GoToステートメント 333

H

[HScrollBar]ボタン 163
HTML
<object>タグ 384
保存 383

I

.ico 341
If...Then...ElseIfステートメント 136
If...Then...Elseステートメント 136
If...Thenステートメント 136
Ifステートメント 69
使用上の注意 122
[Image]ボタン 37
Input関数 214
[Interval]プロパティ 152

K

Killステートメント 230

L

[Label]ボタン 23
[LargeChange]プロパティ 164, 169
Launch.exe 340
[Left]プロパティ 68
Line Input#ステートメント 205
[Line]ボタン 140
Lineメソッド 79
[ListBox]ボタン 177
[ListCount]プロパティ 194
[ListIndex]プロパティ 194
[List]プロパティ 177, 187
LoadPicture関数 44, 154
Loadステートメント 115
LOF関数 214

M

[MaxLength]プロパティ 163
[Max]プロパティ 164, 169
[Min]プロパティ 164, 169
MkDirステートメント 230
[MMControl]ボタン 283
MouseDownイベント 66
MouseUpイベントと組み合わせる 76
MouseMoveイベント 91
MouseUpイベント 72
MouseDownイベントと組み合わせる 76
MSDNライブラリ xxvi
MsgBox関数 11, 192
ボタンタイプと指定値 193
戻り値 194
MSVBVM60.dll 360
[MultiLine]プロパティ 200

N

Not演算子 206
[Number]プロパティ 335

O

<object>タグ 384
.ocx 379
[OLEDragMode]プロパティ 222
On Errorステートメント 270, 332
Openステートメント 205, 214
[OptionButton]ボタン 147

P

[Path]プロパティ 224
[Pattern]プロパティ 223
[Picture]プロパティ 37
PopupMenuメソッド 106, 131
Print#ステートメント 214, 216
Printメソッド 327
Privateプロシージャ 11
Psetメソッド 92

Q

QBColor関数 92, 94

R

[RecordsetType]プロパティ 294
[RecordSource]プロパティ 294
[RecordVisible]プロパティ 284
RemoveItemメソッド 194, 250
RGB関数 92
RmDirステートメント 230

S

SaveSettingステートメント 251
[ScrollBar]プロパティ 200
Select Caseステートメント 183
Shell関数 255
Windows定数値一覧 256
ShowColorメソッド 277
ShowFontメソッド 277
ShowHelpメソッド 277
ShowOpenメソッド 277
ShowPrinterメソッド 277
ShowSaveメソッド 277
Showメソッド 133
[Slot]ボタン 378
[SmallChange]プロパティ 169
[SSTab]ボタン 300
[Stretch]プロパティ 41, 304
Subステートメント 11

T

[TabOrientation]プロパティ 306
[TextBox]ボタン 162
[Text]プロパティ 184, 187
Timerイベント 152
[Timer]ボタン 146

[ToolTipText] プロパティ 163, 165
[Top] プロパティ 68

U

Unloadステートメント 121, 134

V

[Value] プロパティ 164, 182
Variant型 74
VB6JP.dll 360
.vbp 18
Visibleプロパティ 109
Visual Basic 6.0
　開発環境 xxii
　概要 x
　画面構成 xxvi
　起動 xxx
　機能 x
　終了 xxx
　初期画面 4
　ファイルの保存 xxxi
　プログラミング xii

W

[WindowState] プロパティ 256
Windowsアプリケーション xiii
Withステートメント 278
Write#ステートメント 214, 219
WWWブラウザ 383

あ

アイコンファイル 341
アニメーション表示プログラム 138

い

イベント xvii, 55
イベント駆動型プログラミング xv
イベントコントロールの移動 62
イベントプロシージャ 10, 59
　Change 167
　Click 56
　DblClick 56
　MouseDown 66
　MouseMove 91
　MouseUp 72
　Timer 152
　宣言部分 11
　変数の使い方 70
イベント名 11
イミディエイトウィンドウ xxix, 327
イメージ
　切り替え 42
　表示 37
イメージコントロール 37
　[Picture] プロパティ 37
　[Stretch] プロパティ 41, 304
　画像の表示 37
イメージビューワーの作成 220
イメージファイル 36
色の設定 92
[色の設定] ダイアログボックス 280
インターネットトランスファコントロール xi

う

ウォッチ 319
ウォッチウィンドウ xxix

え

エラー 319
エラー処理 328
エラー処理ルーチン 334
エラートラップモード 334
エラー番号 335
エラー番号一覧 410
演算子
　Not演算子 206
　比較演算子 121
演算子（続き）
　文字列連結演算子（&） 73
　論理演算子 168

お

お絵描きプログラムの作成 82
オブジェクト xvii, 4
　Err 270
　整列 367
　選択 6
　名前の変更 32
　フォントの指定 26
オブジェクトブラウザ xxix
オブジェクト名 11, 238
オプションボタンコントロール 147
オンラインヘルプ xxvi

か

[開始] ボタン 8
階層メニューの作成 101
画像
　切り替え 42
　表示 37
カラーパレット xxix
関数 xvii, 11, 398
　EOF 206
　FileDateTime 214
　FileLen 214, 332
　FreeFile 214
　GetSetting 246
　Input 214
　LoadPicture 44, 154
　LOF 214
　MsgBox 11, 192
　QBColor 92, 94
　RGB 92
　Shell 255
関数リファレンス 398

き

キャビネットファイル 357
キャリッジリターン 207

く

区切り線（セパレータ） 100
グラフィック作成ソフト 159
クリックイベント 56

こ

コード xvii
　サウンドファイルの再生 288
コードウィンドウ xxix, 8
　オブジェクト一覧 12
　ショートカット 20
　表示方法 12, 81
　プロシージャ一覧 12
　プロシージャの切り替え 58
コードエディタ xxiii
コードエディタウィンドウ 8
コマンドボタン
　名前の変更 6
　メッセージの表示 2
コマンドボタンコントロール 5
　[ToolTipText］プロパティ 165
コミュニケーションコントロール xi
コメント 96
コモンダイアログボックス
　コントロール 273
コモンダイアログボックスの作成 272
コンテナコントロール
　フレームコントロール参照
コントロール xvii, 6
　ActiveXコントロール xi
　イメージ 37
　オプションボタン 147
　重ねて配置 49
　組み合わせて使用する 46
　コピーと貼り付け 88
コントロール（続き）
　細かく移動 142
　コマンドボタン 5
　コモンダイアログボックス 273
　コンボボックス 187
　サイズ変更 51
　スクロールバー 163
　整列 120
　タイマー 146
　タブ付きダイアログボックス 300
　チェックボックス 176
　ディレクトリリストボックス 222
　データ 293
　テキストボックス 162
　ドライブリストボックス 221
　標準コントロール xi
　ファイルリストボックス 222
　フォーム上に配置 13
　フレーム 47
　ポップアップ用 127
　マルチメディア 283
　メニュー 110
　ライン 140
　ラベル 23
　リストボックス 177
コントロールツールボックス xxviii
コントロール配列 113, 141, 367
コンパイラ xxii
コンパイル 338
コンポーネント 273, 283, 300
コンボボックスコントロール 187
　[ListCount］プロパティ 194
　[ListIndex］プロパティ 194
　[List］プロパティ 187
　[Sorted］プロパティ 192
　[Text］プロパティ 187
コンボボックスの追加 187

さ

サイズ変更ハンドル 6
サウンドファイルの再生 288
参照
　プロパティ 22
[参照］ボタン 349

し

実行ファイル（EXE） 3
　作成 338
　保存 340
自動インクリメント 344
消去
　描画した線 93
状況感知型メニューの作成 124
条件分岐 136
ショートカット
　コードウィンドウ 129
ショートカット一覧 412
[書式］メニュー 120

す

スクロールバーコントロール 163
　Changeイベントの発生 170
　[LargeChange］プロパティ 164, 169
　[Max］プロパティ 164, 169
　[Min］プロパティ 164, 169
　[SmallChange］プロパティ 169
　[Value］プロパティ 164
スクロールバーの追加 163
[スタート］メニュー 3
スタンドアローンプログラム xi
ステートメント xvii, 45
　Call 208
　ChDir 230
　Close 208, 214
　DeleteSetting 250
　Dim 75
　Do While...Loop 206
　Else 122
　End 14, 132
　GoTo 333

ステートメント（続き）
If 69, 121
If...Then 136
If...Then...Else 136
If...Then...ElseIf 136
Kill 230
Line Input# 205
Load 115
MkDir 230
On Error 270, 332
Open 205, 214
Print# 214, 216
RmDir 230
SaveSetting 251
Select Case 183
Sub 11
Unload 121, 134
With 278
Write# 214, 219
エラー処理 328
複数行に分割 243
ステップアウト 322
ステップイン 321
ステップ実行 319
スロットマシンコントロールの作成 364

せ

セットアッププログラム 357
セパレータ（区切り線） 100
宣言
記述場所 75
データ型 74
変数 73

そ

添字 116
属性 22
ソフトウェアランチャーの作成 231

た

タイマーコントロール 146
[App.Path] プロパティ 154
[Enabled] プロパティ 155
[Interval] プロパティ 152
タブ付きダイアログボックス
作成 298
タイプ 307
タブ位置の指定 306
プロパティページ 307
タブ付きダイアログボックス
コントロール 300
[TabOrientation] プロパティ 306
ダブルクリックイベント 56

ち

チェックボックスコントロール 176
[Enabled] プロパティ 183
[Value] プロパティ 182
チェックボックスの追加 176

つ

ツールチップ 171
ツールバー xxviii
作成 138
ボタンの作成 142

て

ディストリビューションウィザード
xxvi, 348
起動 349
指定のやり直し 353
ディレクトリリストボックス 260
追加 222
ディレクトリリストボックス
コントロール 222
[Path] プロパティ 224
データ型一覧 409
データコントロール xi, 292
[DatabaseName] プロパティ 293
[RecordsetType] プロパティ 294
[RecordSource] プロパティ 294
データビューウィンドウ xxix
データベースファイルへのアクセス 292
テキストファイルの読み込み 198
テキストボックスコントロール 162
[MaxLength] プロパティ 163
[MultiLine] プロパティ 200
[ScrollBar] プロパティ 200
[ToolTipText] プロパティ 163
テキストボックスの追加 200
デバッグ xxv, 319
ウォッチ式の追加 318
ステップアウト 322
ステップイン 321
ブレークポイント 324

と

ドライブリストボックス 260
追加 221
ドライブリストボックスコントロール 221
[Drive] プロパティ 223
[Path] プロパティ 223
エラーの回避 270
ドラッグ 5

は

バージョン情報 344
作成 308
バージョン番号 343
背景
透明にする 53
背景色の指定 24
配列 116
添字 116
バグ 319
パス 45
パターン 223

ひ

比較演算子 121
引数 67
描画した線の消去 93
描画プログラム 82
標準.EXE 3
標準コントロール xi

ふ

ファイル
　シーケンシャル書き出し処理 215
　閉じる 208
　中身の取り出し 205
　保存 xxxi
　読み込み 198
　リスト項目の書き出し 210
ファイル操作 203
　関数一覧 214
　ステートメント一覧 214
ファイル番号 203
ファイルリストボックス 260
　[Filename] プロパティ 224
　[OLEDragMode] プロパティ 222, 226
　[Pattern] プロパティ 223
　追加 222
ファイルリストボックスコントロール 222
[ファイルを開く] ダイアログボックス 277
フォーム xvii, 5
　起動場所の変更 9
　選択 38
　他のフォームの操作 156
　有効期限 172
フォーム作成ウィンドウ xxix
フォームテンプレート 308
　情報の変更 310
フォームモジュールの追加 299
フォームレイアウトウィンドウ xxix, 315
　フォームの配置 148
フォント 26
[フォントの指定] ダイアログボックス 279
ブレークポイント 324
　解除 326
　設定 325
フレームコントロール 47, 96
プログラミング xii
　Visual Basic 6.0の開発環境 xxii
　イベント駆動型プログラミング xv
プログラム
　コードの入力 8
　作成 2
　実行 8
　終了 8
　配布キットの作成 348
プロシージャ xvii, 10
　Private 11
　新規作成 204
プロジェクト xvii, 4
　解放 17
　新規作成 3
　閉じるボタン 20
　開く 19
　保存 18
プロジェクトエクスプローラ xxix
プロジェクトファイル xxiv, 4
　拡張子一覧 412
　閉じる 17
プロジェクトフォーム
　名前の確認 263
プロジェクトプロパティ 312
プロパティ xvii, 22
　.(ピリオド) 35
　値 22
　値の代入 30
　参照 22
　指定方法 35
プロパティウィンドウ xxix, 6
プロパティページ 307

へ

変数 xvii, 67
　Variant型 74
　ウォッチ 318
　宣言 73
　使い方 70
　データ型 74
　名前の規則 74

ほ

ポップアップメニュー
　作成 104
　マウスボタンへの割り当て 108

ま

マウスイベント 62, 82
マウスで直線を引く 76
マウスポインタ 5
マルチメディアコントロール xi, 283
　Closeコマンド 287
　[DeviceType] プロパティ 285
　[EjectVisible] プロパティ 284
　Ejectコマンド 287
　Openコマンド 287
　Pauseコマンド 287
　Playコマンド 287
　[RecordVisible] プロパティ 284
　Soundコマンド 287
　Stopコマンド 287
　コードでのサウンドファイルの再生 288
　配置 284

め

メソッド xvii, 79
　AddItem 191
　Cls 93
　Line 79

メソッド（続き）
PopupMenu 106, 131
Print 327
Pset 92
RemoveItem 194, 250
Show 133
ShowColor 277
ShowFont 277
ShowHelp 277
ShowOpen 277
ShowPrinter 277
ShowSave 277
メッセージボックス 192
タイプ 193
メディアの操作 282
メニュー xxviii
階層メニューの作成 101
区切り線の追加 100
グレー表示（使用不可） 117
削除 118
状況感知型メニューの作成 124
追加 110
フォームへの組み込み 98
ポップアップメニューの作成 104
メニューエディタ 99
インデックス番号 113
階層メニューの作成 101
キャプション 99
区切り線（セパレータ） 100
コントロール名 99
状況感知型メニューの作成 124
ポップアップメニューの作成 104
メニューコントロール 110
[Enabled] プロパティ 117
[Visible] プロパティ 124
ダイアログボックスの表示 159

も

モーダル表示 133
モードの確認 326
モードレス表示 133
文字
フォントの指定 26
モジュール xvii, 78
モジュールプログラミング xxiv
文字列連結演算子（&） 73

ゆ

ユーザーインターフェイス x
ユーザーコントロール
コードウィンドウの表示 372
ユーザー定義プロシージャ 204

よ

用語集 392

ら

ラインコントロール
[BorderColor] プロパティ 140
直線の引き方 140
ラインフィード 207
ラベル 23
ラベルコントロール 23
[Alignment] プロパティ 27
[BackColor] プロパティ 24
[BackStyle] プロパティ 53
[BorderStyle] プロパティ 71
[DataField] プロパティ 296
[DataSource] プロパティ 296
[FontItalic] プロパティ 185
[Font] プロパティ 26
[ForeColor] プロパティ 25
[Left] プロパティ 68
[Top] プロパティ 68
背景を透明にする 53
文字の配置 27
ランタイムDLL 337, 360

り

リストボックス
項目の削除 251
追加 177
リストボックスコントロール 177
[List] プロパティ 177
[Text] プロパティ 184

る

ループ処理 206, 208

れ

レジストリ 243
レジストリエディタ 247
連結コントロール 293
練習問題の解答 414

ろ

ローカルウィンドウ xxix
論理演算子 168

● 著者紹介 ●

瀬戸 遥（せと はるか）

フリーソフトの作者からパソコン関係のフリーライターに転身。主に、パソコン通信・ホームページ作成とVisual Basic関連の解説書を中心に執筆、これまで約40点の書籍を翔泳社他多くの出版社より出版している。

現在、月刊「Visual Basic Magazine」（翔泳社）に、「Excel VBA」のテクニック解説記事を連載中。

●カバー装丁　株式会社アレフ・ゼロ（宇田 俊彦）

● 「10日でおぼえる」のホームページ ●

本書の内容確認には十分気を付けておりますが、万一プログラムコードや本文の誤記などで訂正が発生した場合には、以下のホームページにて訂正内容をお知らせします。ご利用下さい。

http://www.shoeisha.com/book/hp/10days/

10日でおぼえる Visual Basic 6.0 入門教室

1998年11月20日　初版第1刷発行
2005年 7月 1日　初版第19刷発行

著　者　VBテックラボ＆瀬戸 遥（せと はるか）
発行人　速水 浩二
発行所　株式会社 翔泳社（http://www.seshop.com）
印　刷　昭和情報プロセス株式会社
製　本　株式会社 国宝社

ISBN4-88135-687-9　Printed in Japan

10日でおぼえる

●ご利用の前に、
本書xxxiiページの
「CD-ROMをご使用の前に」、
およびCD-ROMに収録の
ファイル「README.TXT」を
必ずお読みください。

SE
SHOEISHA

COMPACT disc

CD-ROM for
Windows95/98/NT4

●本CD-ROMに
Visual Basic 6.0は
収録されておりません。

Visual Basic 6.0 入門教室

VBテックラボ
&
瀬戸 遥 著

ISBN4-88135-687-9

C3055 ¥2800E

9784881356876

株式会社 翔泳社

定価：本体 2,800円＋税

1923055028005

第1日 はじめてのプログラムを作る

- 1時限目 コマンドボタンでメッセージを表示するetc
- 2時限目 プロパティを操作するetc
- 3時限目 イメージファイルを使うetc
- 4時限目 コントロールを組み合わせるetc
- 5時限目 自習

第2日 イベントを使う

- 1時限目 クリックとダブルクリック
- 2時限目 マウスイベントを使う
- 3時限目 変数を使う
- 4時限目 MouseDownとMouseUpの組み合わせ
- 5時限目 お絵描きプログラムの作成

第3日 メニューを使う

- 1時限目 メニューを組み込む
- 2時限目 右ボタンポップアップメニューを作る
- 3時限目 メニューを追加する
- 4時限目 メニューの削除
- 5時限目 状況感知型メニューを作成する

第4日 コントロールの連携と動作

- 1時限目 アニメーション表示プログラム
- 2時限目 スクロールバーとツールチップを使う
- 3時限目 チェックボックス・リストボックスコントロールを使う
- 4時限目 コンボボックスでのリスト項目の追加と削除
- 5時限目 自習

第5日 ファイルを操作する

- 1時限目 テキストファイルを読み込む
- 2時限目 リスト項目をファイルに書き出す
- 3時限目 イメージビューワーの作成
- 4時限目 自習
- 5時限目 自習

第6日 ソフトウェアランチャーの作成

- 1時限目 メインモジュールの作成
- 2時限目 セットアップモジュールの作成
- 3時限目 自習
- 4時限目 自習
- 5時限目 自習

第7日 いろいろなテクニック

- 1時限目 コモンダイアログボックス
- 2時限目 マルチメディアを使うetc
- 3時限目 データコントロール
- 4時限目 タブ付きダイアログボックス
- 5時限目 フォームテンプレートの活用

第8日 デバッグとエラー処理

- 1時限目 変数のウォッチを使う
- 2時限目 ブレークポイントを使う
- 3時限目 エラー処理を使う
- 4時限目 自習
- 5時限目 自習

第9日 実行ファイルの作成と配布

- 1時限目 EXEファイルの作成
- 2時限目 バージョンと作成情報を設定する
- 3時限目 ディストリビューションウィザード
- 4時限目 自習
- 5時限目 自習

第10日 ActiveXコントロールを作る

- 1時限目 スロットマシンコントロールの作成
- 2時限目 ActiveXコントロールの組み込みとテスト
- 3時限目 ActiveXコントロールをHTMLに組み込む
- 4時限目 自習
- 5時限目 自習

10日でおぼえる Visual Basic 6.0 入門教室

VBテックラボ＆瀬戸遥 著

SE SHOEISHA